

Oracle OLAP

アプリケーション開発者ガイド

リリース 2 (9.2.0.4)

2003 年 12 月

部品番号 : J08246-01

ORACLE®

Oracle OLAP アプリケーション開発者ガイド, リリース 2 (9.2.0.4)

部品番号: J08246-01

原本名: Oracle OLAP Application Developer's Guide Release 9.2.0.4.1

原本部品番号: B10333-01

Copyright © 2002, 2003, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

*オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに.....	xix
対象読者	xix
このマニュアルの構成	xx
関連文書	xxii
表記規則	xxii

第 I 部 基礎

1 概要

Oracle Database 内の OLAP テクノロジ	1-1
2 つの個別のシステムをメンテナンスする問題点.....	1-1
多次元テクノロジの完全統合	1-2
OLAP を使用したビジネス上の問題解決	1-2
一般的な分析アプリケーション	1-3
OLAP オプションの構成要素	1-3
アナリティック・ワークスペースを使用するタイミングの決定	1-4
プロセスの概要: アナリティック・ワークスペースの作成およびメンテナンス.....	1-5
多次元オブジェクトへのソース・データのマッピング	1-6
OLAP カタログ・メタデータについて	1-6
アナリティック・ワークスペースの作成	1-7
BI Beans および OLAP API のサポート.....	1-7
アナリティック・ワークスペースでの作業用ツール	1-7
Oracle Enterprise Manager	1-9
Analytic Workspace Manager	1-10
OLAP Worksheet	1-10

Oracle Warehouse Builder	1-10
SQL パッケージ.....	1-10

2 多次元データ・モデル

論理多次元データ・モデル	2-1
論理キューブ	2-2
論理メジャー	2-2
論理ディメンション	2-3
論理階層およびレベル	2-3
論理属性	2-4
モデルのリレーショナルな実装	2-4
ディメンション表	2-5
ファクト表	2-6
マテリアライズド・ビュー	2-6
モデルのアナリティック・ワークスペースの実装	2-6
アナリティック・ワークスペースにおける多次元データの格納	2-7
データベース・スタンダード・フォームのアナリティック・ワークスペース	2-8
アナリティック・ワークスペースのディメンション	2-9
スタンダード・フォームのアナリティック・ワークスペースにおける	
ディメンションの使用	2-10
アナリティック・ワークスペースのリレーション	2-10
アナリティック・ワークスペースの変数	2-11
変数を使用したメジャーの格納	2-11
変数を使用した属性の格納	2-12
アナリティック・ワークスペースの式	2-12

3 サンプル・スキーマ

事例のシナリオ	3-1
要件のレポート	3-2
ビジネスの目標	3-2
情報の要件	3-3
ビジネス分析の問合せ	3-3
どの製品が利益が高いか？	3-3
顧客は誰か、彼らは何をどのように購入するのか？	3-4
どのアカウントが最も収益性が高いか？	3-4

各流通チャネルの業績は？	3-4
ビジネスに季節的な分散はまだ存在するか？	3-5
情報の要件のまとめ	3-5
必要とされるビジネス・ファクトの識別	3-5
Global Computing のための論理データ・モデルの設計	3-6
ディメンションの識別	3-6
レベルの識別	3-6
階層の識別	3-7
ストアド・メジャーの識別	3-7
Global スター・スキーマ	3-8
ディメンション表: TIME_DIM	3-9
ディメンション表: CUSTOMER_DIM	3-11
ディメンション表: PRODUCT_DIM	3-12
ディメンション表: CHANNEL_DIM	3-12
ファクト表: UNITS_HISTORY_FACT および UNITS_UPDATE_FACT	3-13
ファクト表: PRICE_AND_COST_HISTORY_FACT および PRICE_AND_COST_UPDATE_FACT	3-14
アナリティック・ワークスペースへの Global スキーマのマッピング	3-14
Global Product ディメンションのマッピング	3-15
Global Time ディメンションのマッピング	3-16
Global Price キューブのマッピング	3-18

4 OLAP 用 Java アプリケーションの開発

分析 Java アプリケーションの作成	4-1
Java の概要	4-1
OLAP の Java ソリューション	4-2
Oracle の Java 開発環境	4-2
BI Beans の概要	4-3
メタデータ	4-4
ナビゲーション	4-4
書式設定	4-4
グラフ	4-4
クロス集計	4-5
表	4-5
データ Beans	4-5
ウィザード	4-5

OLAP API の理解	4-6
OLAP API による多次元データへのアクセス方法	4-6
インテリジェント・キャッシング	4-7
計算機能	4-7

第 II 部 アナリティック・ワークスペースの作成および使用の基礎

5 OLAP カタログ・メタデータの作成

OLAP カタログの概要	5-1
OLAP カタログ・コンポーネント	5-2
CWM1 について	5-2
CWM2 について	5-3
OLAP メタデータを作成するための手順	5-3
OLAP カタログ・メタデータの作成用ツールの選択	5-3
ソース・データのメタデータの作成	5-3
基本的なスター・スキーマまたはスノーフレイク・スキーマ	5-5
複雑な階層を持つディメンション表	5-5
その他のスキーマ構成	5-5
アナリティック・ワークスペースのメタデータの作成	5-6
Oracle Enterprise Manager を使用したメタデータの作成	5-8
手順: OLAP 管理へのアクセス	5-8
ディメンション表へのメタデータの定義	5-9
ディメンションに指定する情報	5-9
時間ディメンション	5-10
手順: OLAP カタログでの論理ディメンションの定義	5-10
ファクト表へのメタデータの定義	5-10
キューブに指定する情報	5-11
手順: OLAP カタログでの論理キューブの定義	5-11
キューブのデータの表示	5-11
事歴: GLOBAL スター・スキーマのメタデータ作成	5-12
Global スキーマの論理 Time ディメンションの定義	5-12
Global スキーマの論理 Units キューブの定義	5-13
PL/SQL を使用したメタデータの作成	5-14
OLAP ディメンションを作成するための CWM2 パッケージ	5-14
キューブを作成するための CWM2 パッケージ	5-14
メタデータをマッピングするための CWM2 パッケージ	5-15

レベルベース・ディメンション表を作成するための CWM2 パッケージ.....	5-15
分類および検証のための CWM2 パッケージ.....	5-15

6 アナリティック・ワークスペースの作成

アナリティック・ワークスペースの作成方法	6-1
Analytic Workspace Manager の概要.....	6-4
OLAP カタログ・ビュー	6-4
オブジェクト・ビュー	6-5
OLAP Worksheet	6-6
Analytic Workspace Manager でデータベース接続を開く	6-7
Analytic Workspace Manager を使用したスタンダード・フォーム・ワークスペースの作成.....	6-7
アナリティック・ワークスペースのスキーマの選択	6-8
拡張記憶域オプションの設定	6-8
コンポジット・ディメンションの定義	6-8
キューブ内のディメンションの順序付け	6-9
セグメント・サイズの設定	6-9
作成オプションの選択	6-10
スクリプトの生成	6-10
スタンダード・フォームのワークスペースを作成する基本手順.....	6-10
事例: Global アナリティック・ワークスペースの作成.....	6-11
GLOBAL_AW ワークスペース・ユーザーの定義.....	6-11
GLOBAL のスパース特性の調査	6-12
アナリティック・ワークスペース作成ウィザードの実行	6-12
オブジェクト定義の手動による変更	6-13
作成の完了	6-15
事例: Sales History アナリティック・ワークスペースの作成.....	6-15
SH 作成用起動パラメータの定義.....	6-15
SH の表領域の定義.....	6-16
SH データのスパース特性の調査.....	6-16
SH 作成の管理.....	6-17
アナリティック・ワークスペース作成ウィザードの実行	6-17
Sales History アナリティック・ワークスペースの作成.....	6-18
集計データの生成	6-18
集計を計算するための方法	6-18
事前集計および格納するレベルの選択方法	6-19

集計プランについて	6-19
集計プランの作成およびデプロイ方法	6-20
集計プランの作成	6-20
集計演算子の変更	6-20
集計プランのデプロイ	6-21
事例: GLOBAL アナリティック・ワークスペースのデータの集計.....	6-21
事前計算のレベルの識別	6-22
Global Price キューブの集計.....	6-23
アプリケーションに対するアナリティック・ワークスペースの有効化	6-23
アナリティック・ワークスペースを有効化する方法	6-24
BI Beans に対する有効化について	6-24
ビューのスター・スキーマ	6-24
アナリティック・ワークスペースの OLAP カタログ・メタデータ.....	6-25
Oracle Discoverer に対してアナリティック・ワークスペースを有効化する方法.....	6-26
Oracle Discoverer に対する有効化について.....	6-26
Discoverer に作成されるビュー	6-26
アナリティック・ワークスペースのデータのリフレッシュ	6-29
リフレッシュ・ウィザードの使用	6-30
異なるリレーショナル表からのリフレッシュ	6-30
事例: Units キューブのリフレッシュ	6-31
データのリフレッシュで再有効化が必要な場合	6-32

7 アナリティック・ワークスペースへの SQL アクセス

SQL アクセスの概要.....	7-1
アナリティック・ワークスペースのデータの操作	7-1
アナリティック・ワークスペースの間合せ	7-3
アクティブ・カタログについて	7-3
カスタム・メジャーのサポート	7-3
カスタム・メジャーの定義方法	7-3
カスタム・メジャーの分析サポート	7-4
予測および回帰	7-4
時系列操作	7-4
財務演算	7-5
統計演算	7-5
数値計算	7-5
テキスト操作	7-5

アロケーション	7-6
集計	7-6
モデル	7-6
DBMS_AW_UTILITIES を使用したカスタム・メジャーの作成	7-7
事例: DBMS_AW_UTILITIES を使用した、Global への Sales の追加	7-7
アナリティック・ワークスペースに関する情報の取得	7-7
DBMS_AW_UTILITIES を使用した、Sales のカスタム・メジャーとしての定義	7-8
ワークスペースの式の表示	7-9
Sales カスタム・メジャーの問合せ	7-10
OLAP_EXPRESSION を使用したカスタム・メジャーの作成	7-10
事例: OLAP_EXPRESSION を使用した、Global への Sales の追加	7-10
OLAP_TABLE を使用した、ワークスペース・データへの直接アクセス	7-11
Oracle のオブジェクト・リレーショナル・モデルについて	7-11
アナリティック・ワークスペースのビューの設計	7-13
プロセスの概要	7-14
オブジェクト型の作成	7-14
オブジェクトの表の作成	7-15
OLAP_TABLE ファンクションの構文	7-15
接続手順	7-15
表型	7-16
OLAP DML コマンド	7-16
制限マップ	7-16
アナリティック・ワークスペース・データのビューの作成	7-17
ビュー用のスクリプトの作成	7-17
事例: OLAP_TABLE を使用した Global カスタム・メジャーの作成	7-18
アナリティック・ワークスペースにおける式の定義	7-18
OLAP_TABLE を使用したアナリティック・ワークスペースの問合せ	7-19
TS_ROW オブジェクトの定義	7-21
OLAP_TABLE ファンクション	7-21
SELECT 文	7-22
OLAP_TABLE を使用した OLAP API 用のビューの作成	7-22
SQL スクリプトの作成および実行	7-22
サンプル・スクリプトについて	7-24
ワークスペース・ビューの OLAP カタログ・メタデータの定義	7-25

8 スタンダード・フォームのアナリティック・ワークスペースの詳細

OLAP ツールを使用して作成されるワークスペースについて	8-1
データベース・スタンダード・フォームについて	8-2
アナリティック・ワークスペースの追加要件	8-3
スタンダード・フォームのディメンション	8-3
Dimdef ディメンション	8-3
アナリティック・ワークスペース・ディメンションの内容	8-4
アナリティック・ワークスペースの Dimdef ディメンションのプロパティ	8-4
ディメンションのスタンダード・フォーム・メタデータ	8-5
ALL_DIMENSIONS ディメンション	8-6
ディメンションの ALL_DESCRIPTIONS 変数	8-6
ディメンションの AW_NAMES 変数	8-6
DIM_LEVELS 値セット	8-6
スタンダード・フォームの階層	8-7
Hierlist ディメンション	8-7
Hierlist ディメンションの内容	8-7
Hierlist ディメンションのプロパティ	8-7
Member_Parentrel リレーション	8-8
Member_Parentrel リレーションの内容	8-8
Member_Parentrel リレーションのプロパティ	8-9
Member_Gid 変数	8-9
Member_Gid 変数の内容	8-9
Member_Gid 変数のプロパティ	8-10
Member_Inhier 変数	8-10
Member_Inhier 変数の内容	8-10
Member_Inhier 変数のプロパティ	8-11
階層のスタンダード・フォーム・メタデータ	8-11
ALL_HIERARCHIES ディメンション	8-11
階層の ALL_DESCRIPTIONS 変数	8-12
DIM_HIERARCHIES 値セット	8-12
DEFAULT_HIER リレーション	8-12
スタンダード・フォームのレベル	8-12
Levellist ディメンション	8-12
Levellist ディメンションの内容	8-12
Levellist ディメンションのプロパティ	8-13
Member_Levelrel リレーション	8-13
Member_Levelrel リレーションの内容	8-13

Member_Levelrel リレーションのプロパティ	8-14
Member_Familyrel リレーション	8-14
Member_Familyrel リレーションの内容	8-14
Member_Familyrel リレーションのプロパティ	8-15
レベルのスタンダード・フォーム・メタデータ	8-15
ALL_LEVELS ディメンション	8-16
レベルの ALL_DESCRIPTIONS 変数	8-16
DIM_LEVELS 値セット	8-16
スタンダード・フォームの属性	8-16
ALL_LANGUAGES ディメンション	8-17
属性のスタンダード・フォーム・メタデータ	8-18
ALL_ATTRIBUTES ディメンション	8-18
属性の ALL_DESCRIPTIONS 変数	8-18
属性の AW_NAMES 変数	8-18
スタンダード・フォームのメジャー	8-19
メジャー変数	8-19
Measuredef 式	8-19
メジャーのスタンダード・フォーム・メタデータ	8-20
ALL_MEASURES ディメンション	8-20
メジャーの ALL_DESCRIPTIONS 変数	8-21
メジャーの AW_NAMES 変数	8-21
CUBE_MEASURES 値セット	8-21
スタンダード・フォームのキューブ	8-21
Cubedef ディメンション	8-21
Cubedef ディメンションの内容	8-21
Cubedef ディメンションのプロパティ	8-22
Comspec 集計マップ	8-23
Loopspec コンボジット・ディメンション	8-24
キューブのスタンダード・フォーム・メタデータ	8-25
ALL_CUBES ディメンション	8-25
キューブの ALL_DESCRIPTIONS 変数	8-25
キューブの AW_NAMES 変数	8-25
CUBE_MEASURES 値セット	8-26
スタンダード・フォームのカタログ	8-26
OLAP API イネーブラのカタログ	8-27
AWCREATE カタログ	8-29

第 III 部 他のソースからのデータ取得

9 スタンダード・フォームのアナリティック・ワークスペースへのメジャーの追加

スタンダード・フォームのアナリティック・ワークスペースでの操作	9-1
OLAP DML コマンドの実行方法	9-2
Analytic Workspace Manager を使用した OLAP DML の実行	9-3
OLAP Worksheet を使用した OLAP DML の実行	9-4
手順: Analytic Workspace Manager から OLAP Worksheet を開く方法	9-4
手順: OLAP Worksheet のエディタの使用	9-5
DBMS_AW.EXECUTE を使用した OLAP DML の実行	9-6
DBMS_AW.EXECUTE コマンドの書式	9-6
SQL からの DML プログラムへの内容の追加	9-6
キューブへのカスタム・メジャーの追加	9-8
スタンダード・フォームのメジャー変数の定義	9-8
式の定義	9-9
新規メジャーの登録	9-11
ALL_MEASURES ディメンション	9-12
ディメンション・メンバーの追加	9-12
アナリティック・ワークスペースへの変更の保存	9-12
ALL_DESCRIPTIONS 変数	9-13
有効なディメンション・メンバー数の制限	9-13
特定のセルの指定	9-14
変数への値の代入	9-15
AW_NAMES 変数	9-15
CUBE_MEASURES 値セット	9-15
事例: Global アナリティック・ワークスペースへのメジャーの追加	9-16
SALES、EXTENDED_COST および MARGIN のメジャーの作成	9-17
GLOBAL への新規変数の作成	9-17
値の計算と変数への格納	9-18
メジャー式の作成	9-19
新規 Global 変数の集計	9-20
GLOBAL へのその他のカスタム・メジャーの追加	9-20
OLAP DML プログラムによる GLOBAL へのメジャーの追加	9-21

10 将来の業績の予測

予測の作成	10-1
-------------	------

予測の作成手順	10-1
予測期間の作成	10-2
結果を格納する変数の定義	10-2
予測プログラムの開発	10-3
予測の生成	10-4
新規キューブの定義	10-4
Cubedef オブジェクトの作成	10-4
デフォルト集計マップの作成	10-5
新規キューブの登録	10-6
ALL_CUBES ディメンションへのキューブの追加	10-6
ALL_DESCRIPTIONS 変数へのキューブの追加	10-6
AW_NAMES 変数へのキューブの追加	10-7
CUBE_MEASURES 値セット内の新規キューブへのメジャーの追加	10-7
手動で作成したキューブに関するトラブルシューティング	10-8
事例 : Global の Sales の予測	10-8
予測メジャー用の新規キューブの定義	10-9
Global の売上の予測メジャーの定義	10-10
Global の売上の予測プログラムの開発	10-11
履歴期間と予測期間の識別	10-11
FORECAST_SALES サンプル・プログラムの引数	10-11
Global の売上の予測データの検討	10-13
予測メジャーの集計と有効化	10-15

11 他のソースからのデータの取得

OLAP データ取得サブシステムの概要	11-1
スタンダード・フォームのアナリティック・ワークスペースを手動で作成する方法	11-2
フラット・ファイルの読み込み	11-3
ファイル読み込みプログラムについて	11-4
ファイル読み込みプログラムの記述	11-5
ワークスペース・オブジェクトへのフィールドのマッピング	11-6
書式指定ファイルの読み込み	11-6
構造化 PRN ファイルの読み込み	11-7
CSV ファイルの読み込み	11-7
メジャーを読み込むためのディメンション・ステータスの設定	11-7
データ・ロードの最適化	11-8
ディメンション・メンバーの読み込みとメンテナンス	11-9

入力値の変換	11-10
基本的な変換	11-10
リレーションを使用したディメンション値の調整	11-10
リレーショナル表からのデータのフェッチ	11-11
OLAP DML の SQL サポート	11-11
プロセス: リレーショナル表からアナリティック・ワークスペース・オブジェクトへの データのコピー	11-12
ディメンション・メンバーの表からのフェッチ	11-13
ディメンション・メンバーのソート	11-14
表からのメジャーのフェッチ	11-15
その他のメタデータ・オブジェクトの移入	11-16
___POP.FMLYREL の使用	11-17
___ORDR.HIERARCHIES の使用	11-17
事歴: 他のソースからの GLOBALX ワークスペースの作成	11-18
GLOBALX スター・スキーマの設計および実装	11-18
GLOBALX スキーマの図	11-18
手順: GLOBALX サンプル・スキーマの作成	11-20
GLOBALX スキーマの OLAP カタログ・メタデータの作成	11-20
GLOBALX アナリティック・ワークスペースの作成	11-21
リレーショナル表からの Price キューブのフェッチ	11-23
GLOBAL.PRODUCT_DIM からの Products のロード	11-24
GLOBAL.TIME_DIM からの Time のロード	11-25
PRICE_AND_COST_HISTORY_FACT からの PRICE キューブのロード	11-28
フラット・ファイルからの Units キューブのロード	11-29
CHANNELS.DAT からの Channels のロード	11-30
CUSTOMERS.DAT からの Customers のロード	11-31
UNITS_CUBE.DAT ファイルの読み込み	11-34
その他のスタンダード・フォームのメタデータ・オブジェクトの移入	11-36
GLOBALX アナリティック・ワークスペースでのツールの使用	11-37

第 IV 部 OLAP のデータベース管理

12 Oracle OLAP の管理

管理の概要	12-1
アナリティック・ワークスペースの表領域の作成	12-2
UNDO 表領域の作成	12-2

アナリティック・ワークスペースの永続表領域の作成	12-3
アナリティック・ワークスペースの一時表領域の作成	12-4
アナリティック・ワークスペースのサイズの問合せ	12-4
ユーザー名の設定	12-5
DBA およびアプリケーション開発者の SQL アクセス	12-5
分析者の SQL アクセス	12-6
OLAP API を使用したデータベース・オブジェクトへのアクセス	12-6
Oracle OLAP の初期化パラメータ	12-7
手順: OLAP 用システム・パラメータの設定	12-7
OLAP_PAGE_POOL_SIZE の設定について	12-8
PGA_AGGREGATE_TARGET の設定について	12-9
OLAP API の初期化パラメータ	12-9
外部ファイルへのアクセスの許可	12-9
データベース・ディレクトリの作成	12-10
データベース・ディレクトリに対するアクセス権の付与	12-10
例: データベース・ディレクトリの作成と使用	12-11
データ記憶域の理解	12-11
ユーザーが所有する表	12-11
システム表	12-12
パフォーマンスの監視	12-13

13 OLAP API 用のマテリアライズド・ビュー

Oracle OLAP でのサマリー管理	13-1
概要と要件	13-2
キューブに必要なマテリアライズド・ビュー	13-2
マテリアライズド・ビューおよび OLAP メタデータ	13-3
例: ディメンションのマテリアライズド・ビュー	13-3
ディメンション階層のマテリアライズド・ビューの作成	13-3
ディメンション階層のビットマップ索引	13-4
ディメンション階層の統計	13-4
例: ファクトのマテリアライズド・ビュー	13-4
ファクトのマテリアライズド・ビューの作成	13-5
ファクトのマテリアライズド・ビューのビットマップ索引	13-6
ファクトのマテリアライズド・ビューの統計	13-6
DBMS_ODM パッケージの使用	13-6

手順: グルーピング・セット形式のマテリアライズド・ビューの作成.....	13-6
例: Sales キューブのグルーピング・セット形式のマテリアライズド・ビューの作成.....	13-7

A OLAP Server からのアップグレード

管理	A-1
管理ツール	A-1
ユーザーの認証	A-2
データ転送	A-2
ローカライゼーション	A-3
アプリケーションのサポート	A-3
プログラミング環境	A-4
通信	A-4
メタデータ	A-5
プログラミング言語の変更	A-5
新しいコマンド	A-5
廃止コマンド	A-5
UPDATE および COMMIT	A-6
Oracle OLAP Server データベースのスタンダード・フォームへの変換	A-6
CREATE_DB_STDFORM の使用対象者	A-6
CREATE_DB_STDFORM で提供される機能	A-7
CREATE_DB_STDFORM では提供されない機能	A-7
Oracle Express Objects メタデータからの変換	A-8
CREATE_DB_STDFORM の構文	A-9
手順: Oracle Express Objects からスタンダード・フォームへの変換	A-9
Time 属性の移入	A-11
ロード・プログラムの変更	A-11
例: スタンダード・フォームへの XADEMO データベースの変換	A-12
スタンダード・フォームの XADEMO アナリティック・ワークスペースの作成	A-12
XADEMO の Time ディメンションについて	A-13
XADEMO Time 属性の移入	A-14

B GLOBALX の作成用プログラム

ユーザーおよび表領域の定義用の SQL スクリプト	B-1
GLOBALX スター・スキーマ用の SQL スクリプト	B-2
OLAP カタログ・メタデータ用の SQL スクリプト	B-4

C アナリティック・ワークスペースのデータベース・スタンダード・フォーム

データベース・スタンダード・フォームの概要	C-1
データベース・スタンダード・フォームの目的	C-1
データベース・スタンダード・フォームの対象読者	C-2
論理モデルとワークスペース・オブジェクト	C-3
キューブの実装	C-3
メジャーの実装	C-3
ディメンションの実装	C-3
ワークスペース・オブジェクトのクラス	C-4
ワークスペース・オブジェクトのプロパティ	C-4
オブジェクトのネーミング規則	C-4
論理名	C-5
ネームスペースの構成	C-5
単純論理名とフルネーム	C-6
ワークスペース・オブジェクトのプロパティ	C-6
実装クラスのオブジェクトに固有のプロパティ	C-6
すべてのワークスペース・オブジェクトのシステム・プロパティ	C-7
すべてのワークスペース・オブジェクトのロール・プロパティ	C-8
実装クラスのオブジェクトのロール・プロパティ値	C-8
カタログ・クラス・オブジェクトのロール・プロパティ値	C-9
機能クラス・オブジェクトのロール・プロパティ値	C-12
拡張クラス・オブジェクトのロール・プロパティ値	C-13
用語: ロール名を使用したオブジェクトの説明	C-14
実装クラスのオブジェクト	C-14
キューブのオブジェクト	C-15
Cubedef ディメンション	C-15
Loopspec コンボジット	C-16
メジャーのオブジェクト	C-17
Measuredef オブジェクト	C-17
COMPSPEC Aggmap	C-17
ディメンションのオブジェクト	C-18
Dimdef ディメンション	C-18
Hierlist ディメンション	C-19
Levellist ディメンション	C-20
Member_Levelrel リレーション	C-21
Member_Parentrel リレーション	C-21

Hier_Levels 値セット	C-22
Attrdef オブジェクト	C-23
カタログ・クラスのオブジェクト	C-24
オブジェクトのリスト	C-24
ALL_CUBES デイメンション	C-24
ALL_MEASURES デイメンション	C-25
ALL_DIMENSIONS デイメンション	C-25
ALL_HIERARCHIES デイメンション	C-25
ALL_LEVELS デイメンション	C-26
ALL_ATTRIBUTES デイメンション	C-27
ALL_OBJECTS デイメンション	C-27
タイプ、ロールおよび言語のリスト	C-28
ALL_OBJTYPES デイメンション	C-28
ALL_DESCTYPES デイメンション	C-29
ALL_ATTRTYPES デイメンション	C-29
AW_ROLES デイメンション	C-29
ALL_LANGUAGES デイメンション	C-30
キューブおよびデイメンションのオブジェクトのリスト	C-31
CUBE_MEASURES 値セット	C-31
DIM_HIERARCHIES 値セット	C-31
DIM_LEVELS 値セット	C-32
DIM_ATTRIBUTES 値セット	C-32
オブジェクト情報のサポート	C-33
AW_NAMES 変数	C-33
AW_COMPSPECS 変数	C-33
AW_LOOPSPECS 変数	C-34
機能クラスのオブジェクト	C-34
ALL_DESCRIPTIONS 変数	C-34
ATTR_INHIER 変数	C-35
DEFAULT_HIER リレーション	C-35
VISIBLE 変数	C-36
Member_Inhier 変数	C-36
Member_Createdby 変数	C-37
Member_Familyrel リレーション	C-37
Member_Gid 変数	C-37
OBJ_CREATEDBY 変数	C-38
OBJ_STATE 変数	C-38

VERSION 変数.....	C-39
拡張クラスのオブジェクト	C-39

用語集

索引

はじめに

『Oracle OLAP アプリケーション開発者ガイド』では、Oracle Database の Enterprise Edition の OLAP オプションを使用して、SQL および Java アプリケーションの分析処理機能を拡張する方法について説明します。

ここでは、次の項目について説明します。

- [対象読者](#)
- [このマニュアルの構成](#)
- [関連文書](#)
- [表記規則](#)

対象読者

このマニュアルは、アプリケーション開発者を対象としています。このマニュアルを参照する場合、SQL の知識および SQL 開発者やデータベース管理者が使用可能な Oracle のツール製品についての一般知識が必要です。

このマニュアルの構成

このマニュアルは、次のように構成されています。

第 I 部「基礎」

第 I 部では、OLAP オプションの基本概念、ツールおよび機能について説明します。

第 1 章「概要」

この章では、OLAP オプションとともにインストールされている Oracle Database で使用できる強力な分析リソースについて説明します。

第 2 章「多次元データ・モデル」

この章では、多次元データ・モデル、および、このモデルをリレーショナル表、アナリティック・ワークスペース、OLAP カタログにどのように実装するのかについて説明します。

第 3 章「サンプル・スキーマ」

この章では、このマニュアルの例に使用されている GLOBAL サンプル・スキーマについて説明します。

第 4 章「OLAP 用 Java アプリケーションの開発」

この章では、OLAP アプリケーションを作成するために使用できる、豊富な機能を持つ開発環境および強力なツールについて説明します。

第 II 部「アナリティック・ワークスペースの作成および使用の基礎」

第 II 部では、グラフィカルなツール・セットを使用してリレーショナル・スキーマからスタンダード・フォームのアナリティック・ワークスペースを作成する手順について説明します。

第 5 章「OLAP カタログ・メタデータの作成」

この章では、OLAP カタログおよび OLAP メタデータでの操作方法を説明します。これによって、データを論理多次元オブジェクトとして定義できるようになります。

第 6 章「アナリティック・ワークスペースの作成」

この章では、Analytic Workspace Manager のウィザードを使用してスタンダード・フォームのアナリティック・ワークスペースを作成する方法について説明します。

第 7 章「アナリティック・ワークスペースへの SQL アクセス」

この章では、アナリティック・ワークスペース内のデータへのアクセスを提供する様々な SQL パッケージについて説明します。

第 8 章「スタンダード・フォームのアナリティック・ワークスペースの詳細」

この章では、スタンダード・フォームのアナリティック・ワークスペースで作成されるオブジェクトについて説明します。

第 III 部「他のソースからのデータ取得」

第 III 部では、スター・スキーマまたはスノーフレイク・スキーマ以外のソースのデータを使用してアナリティック・ワークスペースを新規作成する方法、または既存のアナリティック・ワークスペースを整備する方法について説明します。

第 9 章「スタンダード・フォームのアナリティック・ワークスペースへのメジャーの追加」

この章では、スタンダード・フォームのアナリティック・ワークスペースに計算済メジャーを永続的に追加する方法について説明します。

第 10 章「将来の業績の予測」

この章では、過去の業績に基づいて将来の結果を予測する方法、およびこの情報を新しいスタンダード・フォームのキューブで利用可能にする方法について説明します。

第 11 章「他のソースからのデータの取得」

この章では、OLAP DML のデータ取得機能について説明します。この機能を使用すると、スター・スキーマまたはスノーフレイク・スキーマ以外のソースからスタンダード・フォームのアナリティック・ワークスペースを作成したり、それらのソースから既存のワークスペースにデータを追加したりできます。

第 IV 部「OLAP のデータベース管理」

第 IV 部では、OLAP オプションに関連する管理作業の実行方法について説明します。

第 12 章「Oracle OLAP の管理」

この章では、OLAP オプションに関連する管理作業について説明し、パフォーマンスに関するヒントを提供します。

第 13 章「OLAP API 用のマテリアライズド・ビュー」

この章では、（アナリティック・ワークスペースではなく）リレーショナル表を使用してデータを格納する際の、OLAP API で使用できるマテリアライズド・ビューを作成する方法について説明します。

付録 A「OLAP Server からのアップグレード」

この付録では、Oracle OLAP Server リリース 6.3 から Oracle OLAP へのアップグレード方法、および両者の主な相違点について説明します。

付録 B「GLOBALX の作成用プログラム」

この付録では、アナリティック・ワークスペースの例を作成するために使用する追加のソース・コードを提供します。

付録 C「アナリティック・ワークスペースのデータベース・スタンダード・フォーム」

この付録では、データベース・スタンダード・フォームのアナリティック・ワークスペースの規則について説明します。

用語集

用語集では、OLAP 固有の用語の定義を示します。

関連文書

詳細は、次のマニュアルを参照してください。

- 『Oracle OLAP リファレンス』
- 『Oracle OLAP 開発者ガイド - Oracle OLAP API』
- OLAP API Javadoc

表記規則

このマニュアルで使用される表記規則は、次のとおりです。

規則	意味
. . .	例における垂直の省略記号は、例に直接関連しない情報が省略されていることを示します。
...	文またはコマンドにおける水平の省略記号は、例に直接関連しない文またはコマンドの一部が省略されていることを示します。
太字	本文中の太字は、本文中または用語集（あるいはその両方）で定義されている用語を示します。
<>	山カッコは、ユーザーが指定する名前を囲んでいます。
[]	大カッコは、任意に選択できるオプション句を囲んでいます。
\$	ドル記号は、Windows ではコマンドプロンプトを、Digital UNIX では Bourne シェル・プロンプトを示します。

第 I 部

基礎

第 I 部では、OLAP オプションの基本概念、ツールおよび機能について説明します。第 I 部の各章を参照すると、Oracle Database 内での OLAP オプションの機能方法、多次元データ・モデルの実装方法、およびアプリケーションで多次元データを使用してその分析機能を向上させる方法について理解できます。

ここでは、次の項目について説明します。

- [第 1 章「概要」](#)
- [第 2 章「多次元データ・モデル」](#)
- [第 3 章「サンプル・スキーマ」](#)
- [第 4 章「OLAP 用 Java アプリケーションの開発」](#)

この章では、OLAP オプションとともにインストールされている Oracle Database で使用できる強力な分析リソースについて説明します。この章では、次の項目について説明します。

- Oracle Database 内の OLAP テクノロジ
- OLAP を使用したビジネス上の問題解決
- 一般的な分析アプリケーション
- OLAP オプションの構成要素
- アナリティック・ワークスペースを使用するタイミングの決定
- プロセスの概要: アナリティック・ワークスペースの作成およびメンテナンス
- 多次元オブジェクトへのソース・データのマッピング
- アナリティック・ワークスペースでの作業用ツール

Oracle Database 内の OLAP テクノロジ

多次元テクノロジーが Oracle Database 内で使用できるようになりました。組織は、多次元の OLAP データベースとリレーショナル・データベースのどちらかを選択する必要はありません。Oracle では、多次元表および分析エンジンをデータベースに統合することによって、Oracle Database の管理性、拡張性、信頼性に加えて、多次元分析機能を提供します。

2 つの個別のシステムをメンテナンスする問題点

1 つのリレーショナル・データベースに多次元テクノロジーを統合することが重要なのは、スタンドアロンの多次元データベースのメンテナンスにコストがかかるためです。追加のハードウェアおよび多次元データベースの特殊な管理ツールの使用に習熟したデータベース管理者 (DBA) も必要です。さらに、スタンドアロンの多次元データベースでは、専用 API を使用するアプリケーションが必要となります。これによって、スタンドアロンの多次元データベースに対して実行可能なアプリケーションの数が大幅に制限されます。それは、これらの API で使用できるアプリケーションが少ないためだけでなく、実行されるすべてのデータ

をリレーショナル・データベースから多次元データベースへ転送する必要があるためです。こうした要件によって、企業は、問合せツールおよびレポート・ツールを 2 セット（リレーショナル・データベースおよび多次元データベースにそれぞれ 1 セットずつ）サポートしなければならない場合があります。

多次元テクノロジーの完全統合

対照的に、OLAP オプションは Oracle Database に完全に統合されています。DBA は、データベースの他のすべてのコンポーネントを管理するために使用する同じツールを使用してこのオプションを管理します。DBA は、データベース操作の最適化の一環として、データの格納および計算に最適な位置を決定できます。リレーショナル・データと多次元データはどちらも単一のアプリケーションでアクセスできます。格納およびメソッドの処理における違いは、このアプリケーション自身および分析者から見えなくすることができます。

OLAP が有効化された Oracle Database によって提供された多次元データの情報豊富なリレーショナル・ビューに対して、現在では SQL ベースのアプリケーションで純粋な SQL を使用できます。OLAP の計算は SQL を使用して問合せを発行できるので、アプリケーション開発者は SQL を活用し、モデリング、予測および what-if 分析が含まれるソフトウェアの分析精度を高めることができます。標準的なレポート・アプリケーションは複雑な多次元計算の結果を表示でき、カスタム集計メンバーおよびカスタム・メジャーなどの非定型の問合せツールは、分析者の計算機能範囲を拡張できます。

OLAP を使用したビジネス上の問題解決

リレーショナル・データベースは、ビジネス上の事柄を追跡するために必要なオンライン・トランザクション処理（OLTP）を提供します。リレーショナル・データベースは、データを効率的に選択、格納および取り出すために設計されており、数 GB のディテール・データを格納する場合に適しています。

リレーショナル・データベースを使用して格納するアクティビティの範囲がますます広がっていることで、その成功を理解できます。その結果、リレーショナル・データベースには、ビジネスに関する重要な情報を生み出す豊富なデータが格納されます。この情報によって、競争が激化する市場において多大な優位性をもたらすことができます。

すべてのレベルの意思決定者がビジネス状況の変化にすばやく対応できるように、使用可能なデータからビジネス上の問題解決を導き出すことが重要です。

標準的なトランザクション問合せでは、「注文 84305 はいつ出荷されたか」という問合せが実行されます。この問合せは、仕事を処理する基本的な手順を反映しています。この問合せは、単純なデータ選択および一意の注文番号によって識別される 1 つのレコード（または、多くても数個の関連レコード）の検索を伴います。どの配達会社を使用したかおよびその注文がどこに出荷されたかなど、すべての追加の問合せもおそらく同じレコードによって答えられます。このレコードは、トランザクションの世界（顧客の発注時に始まり、その注文の出荷および支払いの完了時に終了する世界）では有益な生涯を遂げます。トランザクションが終了した時点で、レコードはアーカイブに移すことができます。

一方、標準的な一連の分析問合せでは、「環太平洋地域の今四半期の売上を、1年前の売上とどのように比較するか、次の四半期の売上げ見込みはどうなるか、売上げ見込みを改善するために変更できる要因は何か、この数字を変更した場合どうなるか」という問合せが実行されます。

これらはビジネス・トランザクションの処理に関する問合せではなく、過去の業績の分析に関する問合せ、および将来の業績を向上させ、競争上のさらなる優位性を提供することによって利益の増加につながる判断を行うための問合せです。分析データベースは意思決定者にとっての水晶球です。彼らが今日、正しい決定を行うことができるかどうかは、将来をどのくらい正確に予測できるかに依存しています。これらの問合せに対する回答を取得するには、単一行の計算、時系列の分析、および集計済の履歴データや現行データへのアクセスを実行します。これには OLAP、つまりオンライン分析処理が必要となります。

一般的な分析アプリケーション

OLAP オプションを使用し、機能性および業績の貴重な向上を実現できる一般的なアプリケーションの例をいくつか示します。

- プランニング・アプリケーションを使用すると、組織は結果を予測できます。プランニング・アプリケーションでは、モデル、予測、集計、アロケーション、シナリオの管理などの予測分析ツールを使用して新しいデータを生成します。このタイプのアプリケーションの例には、企業の予算および財務分析システムや、需要計画システムがあります。
- 予算および財務分析システムを使用すると、組織は過去の業績の分析、収支計画の構築、収益目標の達成、財務計画の変更による効果のモデリングができます。管理によって、予測される収益レベルに適した支出および投資レベルを判断できます。財務分析者は、通貨変動などの要因を考慮して、予算および投資の代替計画を作成できます。
- 需要計画システムを使用すると、組織は販売履歴、販売促進計画、価格モデルなどの要因に基づいて、市場の需要を予測できます。組織は、製品需要を予測する様々なシナリオをモデリングし、適切な製造目標を決定できます。

この説明で取り上げているように、分析問合せとトランザクション問合せでは、回答に必要なデータ処理が基本的に異なります。ユーザー、目標、問合せおよび必要なデータの型が異なります。OLAP オプションで強化されたリレーショナル・データ・ウェアハウスでは、データ分析のための最適な環境が提供されます。

OLAP オプションの構成要素

Oracle Database とともにインストールされる OLAP オプションには、次の構成要素が含まれます。

- **OLAP 分析エンジン**は、多次元データの選択および高速計算をサポートしています。個々のセッションは一連の問合せをサポートする状態のまま（通常の分析アプリケーション）であり、ある問合せの出力を次の問合せの入力として簡単に使用できます。

データ操作ツールの包括的なセットでは、モデリング、集計、アロケーション、予測および what-if 分析をサポートします。OLAP エンジンは、Oracle カーネルで実行されます。

- **アナリティック・ワークスペース**は、OLAP エンジンによって操作可能な多次元形式でデータを格納します。アナリティック・ワークスペースは、LOB 表としてリレーショナル・スキーマに格納されます。1つのデータベースに多数のアナリティック・ワークスペースを作成し、ユーザー間で共有できます。リレーショナル・スキーマと同様に、特定のユーザー ID がアナリティック・ワークスペースを所有し、他のユーザーにはそのアナリティック・ワークスペースに対するアクセス権が付与されます。個々のユーザーがアナリティック・ワークスペースへの変更を個別にコピーして保存できるため、アナリティック・ワークスペース環境は、特にプランニング・アプリケーションに対して効果があります。
- **OLAP に対する SQL インタフェース**は、SQL によるアナリティック・ワークスペースへのアクセスを提供します。SQL インタフェースは、Oracle のオブジェクト・テクノロジーを使用する PL/SQL パッケージに実装されています。
- **OLAP DML**は、アナリティック・ワークスペースの作成、データ・コンテナの定義、およびこれらのコンテナに格納されたデータの操作のための、データの定義および操作言語です。OLAP DML を使用すると、多次元オブジェクトの作成と移入、データを解決する規則の定義、および OLAP オプションによってサポートされているすべてのタイプの分析を実行できます。アナリティック・ワークスペースで使用可能なツールの多くは、OLAP DML コマンドを発行します。
- **OLAP カタログ**は、キューブ、メジャー、ディメンションおよび属性などの多次元用語でデータを記述する読み込み API および書き込み API のセットから構成されます。
- **OLAP API**は、OLAP アプリケーションの Java ベースのプログラミング・インタフェースで、BI Beans をサポートしています。BI Beans は、Java で分析アプリケーションを開発するためのビルディング・ブロックで、JDeveloper で使用できます。Java 開発者は、分析アプリケーションに BI Beans を使用することを検討してください。BI Beans は OLAP オプションには含まれておらず、OLAP が有効化された Oracle Database を必要とすることに注意する必要があります。

このマニュアルでは、SQL アプリケーション開発者にとってすでに使用可能な分析機能を大幅に強化する、アナリティック・ワークスペースを使用したデータの格納および操作に関して重点的に説明します。

アナリティック・ワークスペースを使用するタイミングの決定

データ・ウェアハウスに対して実行されるアプリケーションによって行われる分析のタイプによって、データをアナリティック・ワークスペースに完全に格納するか、アナリティック・ワークスペースとリレーショナル表間に分散するかを決定できます。

アナリティック・ワークスペースは、集計データの生成または格納に関してマテリアライズド・ビューにかわるものです。アナリティック・ワークスペースは、マテリアライズド・ビューでは使用できない、加重計算、非加算的なメソッドおよびモデルなどの複雑な集計メ

ソッドを提供します。集計データに関する格納に問題がある場合にも、アナリティック・ワークスペースを選択する場合があります。アナリティック・ワークスペースは、データが完全に事前集計されているか、部分的に事前集計されているか、またはオンデマンドで完全に集計されるかにかかわらず、アプリケーションに対して完全に解決したデータを常に表示します。OLAP 集計システムの柔軟性によって、実行時の応答時間を妥協せずに、データ・リフレッシュ・ウィンドウの制限内で事前集計することができます。また、アナリティック・ワークスペースは、事前集計されたデータを非常に効率よく格納できます。

モデル、予測および what-if シナリオなどの予測分析ファクションをサポートするアプリケーション用に、アナリティック・ワークスペースの使用を選択する場合があります。さらに、アナリティック・ワークスペースは、実行時に計算されカスタム・メジャーをサポートできる単一行の計算の実行に対して高度に最適化されています。

分散ソリューションは、アナリティック・ワークスペースの拡張計算機能をあまり使用しない問合せアプリケーションおよびレポート・アプリケーションに最適ともいえます。このようなタイプのアプリケーションには、結果が分析者に直接送られるカリレーショナル表に書き込まれる、より集約型の分析用にアナリティック・ワークスペースを実行時に作成および移入することができます。分散モデルの実装は、すべてのデータをリレーショナル表に格納するソリューションから、すべてのデータをアナリティック・ワークスペースに格納するソリューションまでの範囲内で多岐にわたるため、幅広く異なります。

プロセスの概要：アナリティック・ワークスペースの作成およびメンテナンス

アナリティック・ワークスペースは、リレーショナル・データベース、Oracle OLAP Server のレガシー・データベースまたはフラット・ファイルなど、データソースに応じて、様々な方法で作成できます。データソースに応じて、これらの手順の一部は自動的に実行される場合があります。ただし、基本的なプロセスはすべてにおいて同じです。

次に基本手順を示しますが、詳細はこのマニュアルの後の章で説明します。

1. アナリティック・ワークスペースで操作するメジャーを識別します。これらのメジャーをサポートするディメンション、階層、レベルおよび属性を識別します（第2章「[多次元データ・モデル](#)」を参照）。
2. これらのディメンション、メジャーおよびその他のオブジェクトに論理スキーマを定義します（第5章「[OLAP カタログ・メタデータの作成](#)」を参照）。
3. アナリティック・ワークスペースを作成および移入します。
 - a. 空のアナリティック・ワークスペース（Oracle Database 内の LOB 表）を作成します。
 - b. 多次元コンテナを定義します。
 - c. 論理スキーマを使用して、ソース・データを多次元コンテナにマップします。
 - d. アナリティック・ワークスペースにデータをロードします。

- e. 集計、アロケーション、予測およびモデリングの手法を使用してデータを解決します。

(第 6 章「アナリティック・ワークスペースの作成」および第 11 章「他のソースからのデータの取得」を参照)

4. ワークスペースをアプリケーションで使えるようにします。つまり、ビューを生成し、メタデータを特定のアプリケーションで使えるよう定義します (第 6 章「アナリティック・ワークスペースの作成」を参照)。
5. 新しいデータを持ったアナリティック・ワークスペースを定期的にはリフレッシュします (第 6 章「アナリティック・ワークスペースの作成」を参照)。
6. カスタム・メジャーを計算します (第 7 章「アナリティック・ワークスペースへの SQL アクセス」および第 9 章「スタンダード・フォームのアナリティック・ワークスペースへのメジャーの追加」を参照)。

アナリティック・ワークスペースが使えるようになったら、アナリティック・ワークスペースに対して次の任意の方法で SQL アプリケーションを実行できます。

- データベース管理者によって提供されているアナリティック・ワークスペース・データのビューに対して、SELECT 文を発行する。
- OLAP_TABLE ファンクションを使用して、アナリティック・ワークスペースに対して SELECT 文を直接発行する。
- アナリティック・ワークスペースに対して OLAP DML コマンドを実行する DBMS_AW PL/SQL パッケージを使用する。

多次元オブジェクトへのソース・データのマッピング

ソース・データを識別したら、それらのソースを論理多次元オブジェクトにマップする必要があります。

OLAP カタログ・メタデータについて

OLAP カタログは、OLAP オプションに提供されるメタデータ・リポジトリです。PL/SQL プロシージャのセットである書込み API および Oracle Database 内のリレーショナル・ビューである読込み API から構成されています。OLAP カタログは、次の 2 つの別個の機能を実行するために使用します。

- スター・スキーマまたはスノーフレイク・スキーマからアナリティック・ワークスペースを作成する。
- BI Beans を使用する Java アプリケーションに、アナリティック・ワークスペースまたはリレーショナル表に格納されているデータへのアクセスを提供する。BI Beans および OLAP API では、OLAP カタログ・メタデータが必要とされます。データが OLAP カタログに定義されていない場合、BI Beans または OLAP API を使用するアプリケーションでデータを使用することはできません。

OLAP カタログの読み込み API は、定義済のメタデータをアプリケーションで使用できるようにします。読み込み API は、アナリティック・ワークスペース・データのビューに対して SQL の SELECT 文を使用するすべてのアプリケーションに対して有効です。

SQL アプリケーションでは、OLAP カタログを使用する必要はありませんが、使用することで役に立つことがあります。SQL アプリケーションは、基になるデータの位置を認識することなく、OLAP カタログに定義されている論理オブジェクトに対して実行できます。

アナリティック・ワークスペースの作成

アナリティック・ワークスペースを作成する際は、Oracle Enterprise Manager の OLAP 管理ツールを使用してメタデータを作成できます。OLAP 管理ツールは、スター・スキーマまたはスノーフレイク・スキーマにメタデータを定義するために使用される OLAP カタログの書き込み API のサブセットに対するグラフィカル・ユーザー・インタフェース (GUI) を提供します。メタデータを作成したら、Analytic Workspace Manager を使用して、OLAP カタログに定義されている論理オブジェクトをインスタンス化するアナリティック・ワークスペースを作成します。

BI Beans および OLAP API のサポート

BI Beans および OLAP API を使用して、リレーショナル表またはアナリティック・ワークスペースに格納されているデータにアクセスすることができます。アナリティック・ワークスペースを作成しない場合、DBMS_ODM PL/SQL パッケージを使用して、マテリアライズド・ビューを生成できます。

アナリティック・ワークスペースを使用する場合は、Oracle Enterprise Manager の OLAP 管理ツールを使用して、リレーショナル・スキーマにメタデータを作成します。次に、マテリアライズド・ビューを生成するかわりに、ワークスペースを作成し、ワークスペース・オブジェクトのリレーショナル・ビューを生成して、それらのビューに対して OLAP カタログ・メタデータを定義します。Analytic Workspace Manager のウィザードを使用して、これらの手順を実行します。

アナリティック・ワークスペースでの作業用ツール

アナリティック・ワークスペースで作業可能なレベルには次の 3 つがあります。

- グラフィカル・ユーザー・インタフェース (GUI) は、スター・スキーマやスノーフレイク・スキーマからアナリティック・ワークスペースを作成し、そのワークスペースをアプリケーションで使用できるようにする際の基本タスクを実行するための、ウィザードおよびプロパティ・シートを提供します。この最上位のレベルによって、基になる SQL パッケージに対するコールが作成されます。初めてアナリティック・ワークスペースを開発およびメンテナンスする場合は、これらの GUI の使用方法を学習します。
- SQL パッケージは、アプリケーションで使用できるようにアナリティック・ワークスペースを作成、メンテナンスおよび表示するのに必要なすべてのタスクを実行します。一部の SQL パッケージはワークスペースに直接機能し、基底の OLAP DML を実行します。

他の SQL パッケージは、リレーショナル表およびビューに機能し、SQL を実行します。SQL スクリプト内でこれらの SQL パッケージをコールすることによって、一部のタスクを自動化することが必要な場合もあります。

- OLAP DML は、アナリティック・ワークスペースにネイティブな完成された低レベルの言語です。これによって、データの取得、操作および分析において最大限の機能と柔軟性が提供されます。Oracle OLAP Server からのアップグレードの場合、または、より高いレベルのツールではサポートされていない形式でデータが現在格納されている場合、OLAP DML を初期段階で直接使用して作業する必要がある場合があります。それ以外では、ワークスペースの機能を向上させるために OLAP DML を使用する場合があります。

表 1-1 に、開発の各段階でアナリティック・ワークスペースでの作業に使用するツールを示します。

表 1-1 アナリティック・ワークスペースでの作業用ツール

段階	ツール
論理モデルを設計し、リレーショナル表の列を多次元オブジェクトにマップするメタデータを作成する。	Oracle Enterprise Manager の OLAP 管理ツール または CWM2 の書込み API または Oracle Warehouse Builder
アナリティック・ワークスペースを作成する。	Analytic Workspace Manager のウィザード または DBMS_AWM PL/SQL パッケージ または Oracle Warehouse Builder
サマリー・データを生成する。	Analytic Workspace Manager のウィザード または DBMS_AWM PL/SQL パッケージ または OLAP DML
アナリティック・ワークスペースのビューを生成する。	Analytic Workspace Manager のイネーブラ または OLAP_TABLE ファンクション

表 1-1 アナリティック・ワークスペースでの作業用ツール

段階	ツール
アナリティック・ワークスペース のビューのメタデータを生成する。	Analytic Workspace Manager のイネーブラ または CWM2 の書き込み API
カスタム・メジャーを生成する。	DBMS_AW_UTILITIES PL/SQL パッケージ または DBMS_AW PL/SQL パッケージ または OLAP_TABLE ファンクション または OLAP DML
データをリフレッシュする。	Analytic Workspace Manager のウィザード または DBMS_AWM PL/SQL パッケージ または OLAP DML

これらのツールの説明を次に示します。

Oracle Enterprise Manager

Oracle Enterprise Manager はシステム管理ツールで、複雑な SQL コマンドを作成せずに Oracle 製品を管理するための統合されたソリューションを提供します。Oracle Enterprise Manager を使用して、ユーザー・アカウントの設定、表領域の定義、パフォーマンスの監視、および OLAP オプションを含む、データベースに関連するその他の管理作業を実行できます。

OLAP 管理ツールは、データ・ウェアハウス・サポート機能として Oracle Enterprise Manager に組み込まれています。GUI を使用して、ディメンションを作成するデータベース要件を満たすスター・スキーマまたはスノーフレーク・スキーマのディメンション表およびファクト表に対して、OLAP カタログに論理メタデータのディメンション、メジャーおよびキューブを定義できます。

OLAP 管理ツールの詳細は、[第 5 章「OLAP カタログ・メタデータの作成」](#)を参照してください。

Analytic Workspace Manager

Analytic Workspace Manager は、データベース・スタンダード・フォームのアナリティック・ワークスペースを作成するためのユーザー・インタフェースを提供します。このスタンダード・フォームによって、アナリティック・ワークスペースは、データを集計、リフレッシュおよび有効化する様々なツールで使用できるようになり、BI Beans や Oracle Discoverer を使用する OLAP アプリケーションでアクセスできるようになります。これらのツールは、Analytic Workspace Manager でも提供されます。

Analytic Workspace Manager の詳細は、[第 6 章「アナリティック・ワークスペースの作成」](#)を参照してください。

OLAP Worksheet

OLAP Worksheet は、SQL*Plus Worksheet と同様、アナリティック・ワークスペースで作業するための対話型の環境です。OLAP Worksheet は OLAP DML によるアクセスが容易で、モデリング、予測およびアロケーションなど、高度なビジネス分析を実行できます。OLAP DML を使用してアナリティック・ワークスペースで作業するモードと、SQL を使用してリレーショナル表およびビューで作業するモードの 2 つの異なるモードを切り替えることができます。OLAP Worksheet は、Analytic Workspace Manager を介して利用できます。

OLAP Worksheet の詳細は、[第 9 章「スタンダード・フォームのアナリティック・ワークスペースへのメジャーの追加」](#)を参照してください。

Oracle Warehouse Builder

Oracle Warehouse Builder は、多数の異なるソースからのデータの抽出、リレーショナル・データベースのスター・スキーマへのそのデータの変換、OLAP カタログ・メタデータの生成、およびアナリティック・ワークスペースの作成を行うことができます。Warehouse Builder は、Enterprise Manager の OLAP 管理ツール、および Analytic Workspace Manager のアナリティック・ワークスペース作成ウィザードのかわりに使用できます。結果としてできたアナリティック・ワークスペースはデータベース・スタンダード・フォームなので、Analytic Workspace Manager を使用して、データを集計、整備および有効化できます。

データを変換する必要がある場合も、Oracle Warehouse Builder は、アナリティック・ワークスペースを生成するために最適な方法を提供します。データ・ウェアハウスの論理モデルを作成したら、Oracle Warehouse Builder では、わずかな追加手順のみでスター・スキーマに加え、アナリティック・ワークスペースを生成できます。

Oracle Warehouse Builder の詳細は、『Oracle Warehouse Builder ユーザーズ・ガイド』を参照してください。

SQL パッケージ

いくつかの SQL パッケージでアナリティック・ワークスペースがサポートされています。次に主要なものを示します。

- CWM2 は、OLAP カタログ・メタデータを定義するための多数のパッケージの集まりです。これらのパッケージは、Analytic Workspace Manager の OLAP API イネーブラ・ウィザードをサポートしています。
- DBMS_AW には、OLAP DML コマンドを実行するためのプロシージャが含まれています。このパッケージでは、OLAP Worksheet、および Analytic Workspace Manager のプロパティ・シートとダイアログ・ボックスがサポートされています。DBMS_AW パッケージのプロシージャおよびファンクションを使用して、SQL プログラマは、アナリティック・ワークスペース・データに対して直接 OLAP DML コマンドを実行できます。SQL プログラマは、リレーショナル表からアナリティック・ワークスペースにデータを移動し、データの高度な分析（たとえば、予測など）を行った後、分析の結果をリレーショナル表に戻すことができます。
- DBMS_AWM には、アナリティック・ワークスペースを作成するためのプロシージャが含まれています。これは、Analytic Workspace Manager のアナリティック・ワークスペース作成ウィザードをサポートしています。
- DBMS_AW_UTILITIES には、スタンダード・フォームのアナリティック・ワークスペースのカスタム・メジャーを作成および管理するためのプロシージャが含まれています。カスタム・メジャーは実行時に定義され、ストアド・メジャーから計算されます。

これらの PL/SQL パッケージの詳細は、[第 7 章「アナリティック・ワークスペースへの SQL アクセス」](#) および『Oracle OLAP リファレンス』を参照してください。

多次元データ・モデル

この章では、多次元データ・モデル、およびこのモデルをリレーショナル表やスタンダード・フォームのアナリティック・ワークスペースにどのように実装するのかについて説明します。この章では、次の項目について説明します。

- 論理多次元データ・モデル
- モデルのリレーショナルな実装
- モデルのアナリティック・ワークスペースの実装

論理多次元データ・モデル

多次元データ・モデルは、オンライン分析処理、つまり OLAP に不可欠です。OLAP はオンラインのため、素早く回答を提供する必要があります。分析者は、一晩中実行されるバッチ・ジョブの中ではなく、対話型セッションにおいて反復問合せを行うからです。また、OLAP は分析であるため、その問合せは複雑です。多次元データ・モデルは、複雑な問合せをリアルタイムで解決するよう設計されています。

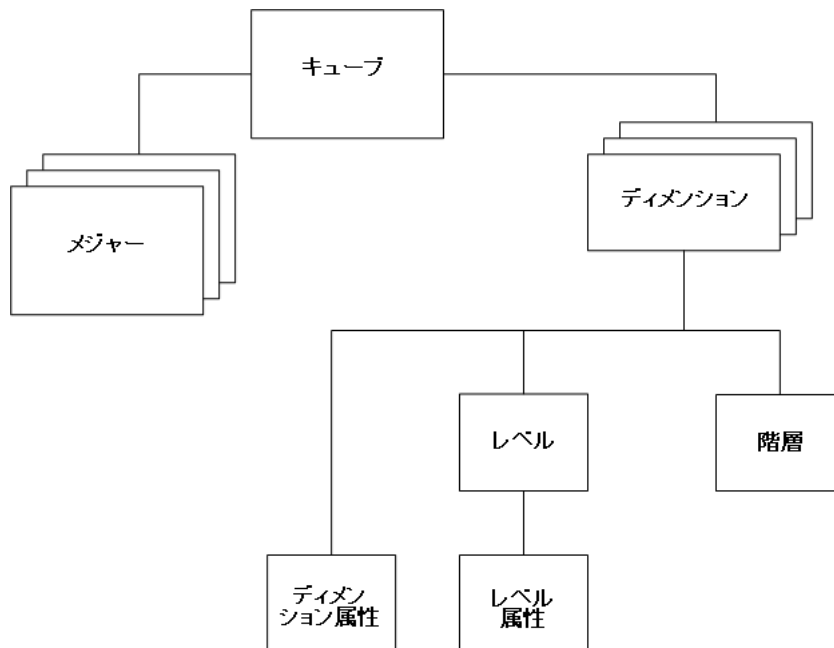
多次元データ・モデルが重要なのは、簡素化を実施しているためです。Ralph Kimball は、彼の著書『The Data Warehouse Toolkit』で次のように述べています。

「ビジネスの多次元モデルの主な魅力は、その簡素性にあります。.... その簡素性こそが、ユーザーがデータベースを理解し、ソフトウェアでデータベースを効率的に操作できるようにする重要な鍵なのです。」

多次元データ・モデルは、論理的なキューブ、メジャー、ディメンション、階層、レベルおよび属性から構成されます。モデルに簡素性が備わっているのは、実在するビジネス・エンティティを表すオブジェクトを定義しているためです。分析者は、調査したいメジャー、データに意味を与えているディメンションや属性、およびビジネスのディメンションをレベルや階層に編成する方法が分かります。

図 2-1 に、論理オブジェクト間の関係を示します。

図 2-1 論理多次元モデルの図



論理キューブ

論理キューブは、同じ形を持つ（つまり、まったく同じディメンションを持つ）メジャーを編成する方法を提供します。同じキューブ内のメジャーは、他の論理オブジェクトに対して同じ関係を持っており、一緒に分析したり表示したりすることが簡単にできます。

論理メジャー

メジャーは、事業活動に関連して収集されたファクトを論理キューブのセルに移入します。メジャーはディメンションによって編成され、通常、これには **Time** ディメンションが含まれます。

分析データベースには、レガシー・システム、トランザクション・データベース、シンジケート・ソースまたはその他のデータソース内のデータから取得された、履歴データのスナップショットが含まれます。通常、3年間分の履歴データが、分析アプリケーションにとって適切と考えられています。

分析者がメジャーを使用して決定を通知している間も、メジャーは静的で一貫性を保ちます。メジャーは、毎週、毎日または一日を通じて定期的に、一定の間隔でバッチ・ウィンドウ内で更新されます。アプリケーションの多くは、メジャーの時間ディメンションにいくつ

かの期間を追加することによってデータをリフレッシュし、最も古い期間を同じ数だけ排除します。各更新によって、その間隔の特定の事業活動の固定された履歴レコードが提供されます。その他のアプリケーションでは、増分更新ではなく、データの完全な再構築を行います。

メジャーの定義では、細目の最低レベル（グレインとも呼ばれます）の決定が重要です。この**ベース・レベル・データ**はユーザーに表示されることはありませんが、これによって、実行可能な分析のタイプが決まります。たとえば、市場分析者は受注担当者とは違い、Michigan 州 Ann Arbor 市在住の Beth Miller によって、サイズ 10 の青色の水玉模様の洋服が 2002 年 7 月 6 日午後 2 時 34 分に注文された、といったことを知る必要はありません。ここで市場分析者が知りたいのは、アメリカ中西部における 2002 年夏の最も人気のあった洋服の色、といった情報です。

この問合せに対する回答を分析者が取得できるかどうかは、このベース・レベルのデータによって決まります。この特定の問合せに対しては、Time は月に、Customer は地域に、Product は色の属性を持つ項目（洋服など）にロール・アップできます。ただし、集計データのこのレベルでは「1 日のうちで女性が最もよく注文するのは何時か」といった問合せに回答できません。データがデータ・ウェアハウスにロードされる前にどの程度までそのデータを事前集計するかが重要な決定となります。

論理ディメンション

ディメンションには、データを識別および分類する一意の値のセットが含まれます。ディメンションによって論理キューブのエッジが形成されます。通常、メジャーは多次元であるため、メジャーの 1 つの値は、意味をなすように各ディメンションのメンバーによって修飾される必要があります。たとえば、Sales メジャーは、Time、Customer、Product および Channel の 4 つのディメンションを持ちます。特定の Sales の値 (43,613.50) は、特定の期間 (Feb-01)、顧客 (Warren Systems)、製品 (Portable PC) およびチャネル (Catalog) によって修飾された場合にのみ意味を持ちます。

論理階層およびレベル

階層は、集計の様々なレベルでデータを構成するための方法です。データを表示する際、分析者はディメンション階層を使用して、あるレベルの傾向を認識し、この傾向の理由を識別するために下位レベルにドリルダウンし、この傾向がビジネスのより広い範囲に与える影響を確認するために上位レベルにロールアップします。

各**レベル**は階層内の位置を表現します。ベース・レベル（最も詳細）より上にある各レベルには、それより低いレベルの集計値が表示されます。レベルが異なるメンバーには、1 対多の**親子リレーション**があります。たとえば、Q1-02 および Q2-02 は 2002 の子、つまり、2002 は Q1-02 および Q2-02 の親です。

階層およびレベルには多対多の関係があります。階層には通常いくつかのレベルが含まれ、1 つのレベルを複数の階層に含めることができます。

1 日に 3 回、つまり、8 時間ごとに取得されるデータのスナップショットがデータ・ウェアハウスに格納されているとします。分析者にとっては通常、日、週、四半期または年に集計

されているデータを表示する方が好ましい場合があります。したがって、Time ディメンションには少なくとも5つのレベルを持つ階層が必要となります。

同様に、来年に特定の目標を持つ販売部長は、自身の販売地域の販売担当者に目標額を割り当てたい場合があります。この割当てでは、個々の販売担当者が特定の地域の子値であるディメンション階層が必要となります。

論理属性

属性は、データに関する追加の情報を提供します。表示用に使用される属性もいくつかあります。たとえば、製品ディメンションで、ディメンション・メンバーに対して最小在庫管理単位 (SKU) を使用する場合があります。SKU は数千の製品を一意に識別する優れた方法ですが、レポートやグラフ内でデータのラベル付けに使用される場合、ほとんどの人にとって意味がありません。こうした説明ラベルには属性を定義します。

属性として、色、味またはサイズなどを持つ場合もあります。このタイプの属性は、「2002年夏の婦人服で最も人気のあった色は」や「この結果は昨年夏とどのように比較されるか」などの問合せに対する、データの選択および回答に使用することができます。

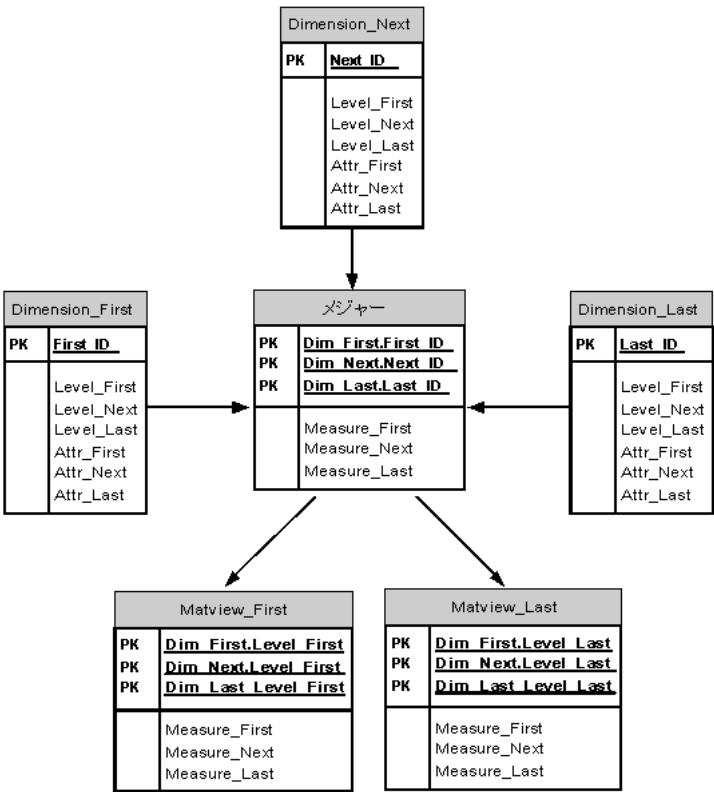
Time 属性は、各期間の最終日や日数の識別など、特定のタイプの分析に役に立つ Time ディメンションに関する情報を提供できます。

モデルのリレーショナルな実装

多次元データ・モデルのリレーショナルな実装を、一般的に**スター・スキーマ**（図 2-2 または**スノーフレーク・スキーマ**を参照）といいます。スター・スキーマは、ディメンション表、ファクト表およびマテリアライズド・ビューにデータを編成するための規則です。究極的にはこれらはすべて表で、メタデータはこれらの表によって表される多次元オブジェクトの識別に必要とされます。

Oracle Database では、OLAP カタログを使用して、リレーショナル表の論理多次元モデルを定義できます。メタデータによって、ディメンション表の属性列とレベル列が区別され、レベル間の階層関係が指定されます。これらの論理オブジェクトの表示名およびメジャーの集計メソッドも提供されます。

図 2-2 スター・スキーマの図



ディメンション表

スター・スキーマには、単一の表内のディメンションに関するすべての情報が格納されます。階層の各レベルは、ディメンション表の1つの列または列セットによって表されます。ディメンション・オブジェクトは、階層の2つのレベルを表す2つの列（または列セット）間の階層関係の定義に使用できます。ディメンション・オブジェクトがない場合、階層関係はメタデータ内でのみ定義されます。属性はディメンション表の列に格納されます。

スノーフレーク・スキーマは、各レベルを個別の表に格納することによって、ディメンション・メンバーを正規化します。

ファクト表

メジャーはファクト表に格納されます。ファクト表には、複数の外部キー（ディメンション表ごとに1つ）から構成されるコンポジット主キー、およびこれらのディメンションを使用する各メジャーの列が格納されます。

マテリアライズド・ビュー

集計データは、ディメンション表に定義された階層関係に基づいて計算されます。これらの集計は、集計表またはマテリアライズド・ビューと呼ばれる個別の表に格納されます。Oracle ではマテリアライズド・ビューに、自動リフレッシュおよびクエリー・リライトなどの幅広いサポートを提供しています。

問合せは、ファクト表またはマテリアライズド・ビューに対して記述できます。ファクト表に対して、結果セットに集計データを必要とする問合せを記述する場合、その問合せは、クエリー・リライトによって既存のマテリアライズド・ビューにリダイレクトされるか、その場でデータが集計されます。

各マテリアライズド・ビューはレベルの特定の組合せに固有です。図 2-2 に示されているマテリアライズド・ビューは、考え得る 27 の組合せ（3 つのレベルを持つ 3 つのディメンションには、3 の 3 乗個のレベルの組合せが考えられる）のうち 2 つのみです。

モデルのアナリティック・ワークスペースの実装

アナリティック・ワークスペースは、ディメンション、変数およびリレーションなど、様々な異なるタイプのデータ・コンテナを持ちます。各タイプのコンテナは、異なるタイプの情報を格納するために様々な方法で使用できます。ディメンション・オブジェクト自体は 1 次元の値のリストであるのに対して、変数およびリレーションは特に、多次元データの効率的な格納、取得および操作をサポートするために設計されています。

注意： アナリティック・ワークスペースは、データベースのデータ・ディクショナリに表として登録されます。ただし、アナリティック・ワークスペースに格納されるオブジェクトは、データベースのデータ・ディクショナリには登録されません。

リレーショナル表同様、アナリティック・ワークスペースには内容に関する明確な要件がありません。空のアナリティック・ワークスペースを作成し、OLAP DML プログラムのみを移入したり、単一のディメンションを定義して値のリストを保持させることができます。ただし、このマニュアルでは、**データベース・スタンダード・フォーム**に準拠するアナリティック・ワークスペースについて説明します。データベース・スタンダード・フォーム（または単にスタンダード・フォーム）は、論理多次元モデルを特定の方法でインスタンス化し、Oracle OLAP ユーティリティの現行のセットで管理できるようにするための規則です。スタンダード・フォームは、特定の論理オブジェクトをインスタンス化するワークスペース・オブジェクトを識別する、アナリティック・ワークスペース内のメタデータを提供

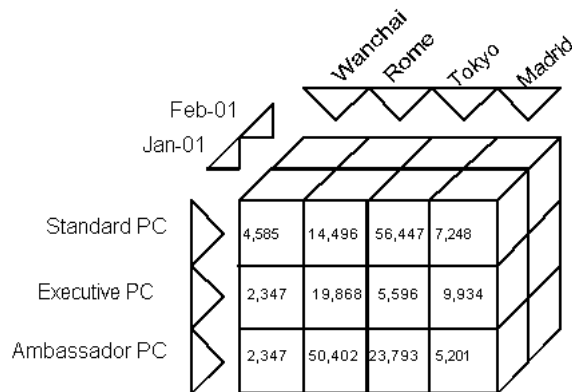
します。スタンダード・フォームについては、このマニュアル全般および付録 C で説明します。

アナリティック・ワークスペースにおける多次元データの格納

キューブの形は、一般的に多次元データの定義に使用される物理的なイメージです。各エッジはディメンションを表します。ディメンション・メンバーはエッジに沿って整列し、キューブの形をデータ値が格納される複数のセルに分割します。

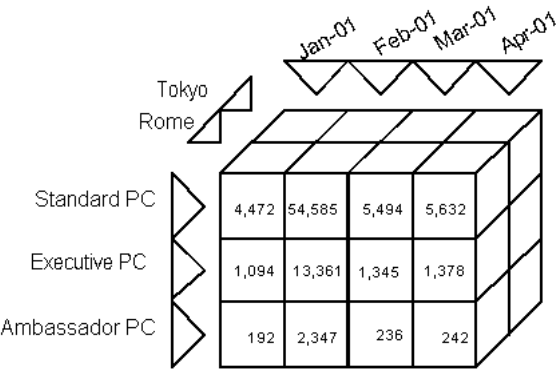
キューブの形は 3 次元です。もちろん、メジャーは 4 つ以上のディメンションを持つことができますが、図で表すことができるディメンションは最高 3 つまでです。追加のディメンションは、通常、キューブを追加して表示します。

会社で売上げデータを収集するとします。会社は、特定の期間に特定の販売地域で各製品がいくつ売れたかを示すレコードを維持しています。この売上メジャーは次のように表示できます。



キューブ形の利点は、データ表示のために回転できることです。すなわち、データを操作または表示するための唯一の正しい方法があるわけではありません。これは多次元モデルの重要な部分です。なぜなら、分析者によって、データを分析および表示する方法がそれぞれ異なるからです。たとえば、環太平洋地域の販売部長は、製品部長や財務分析者とは異なる視点でデータを調べる必要があります。

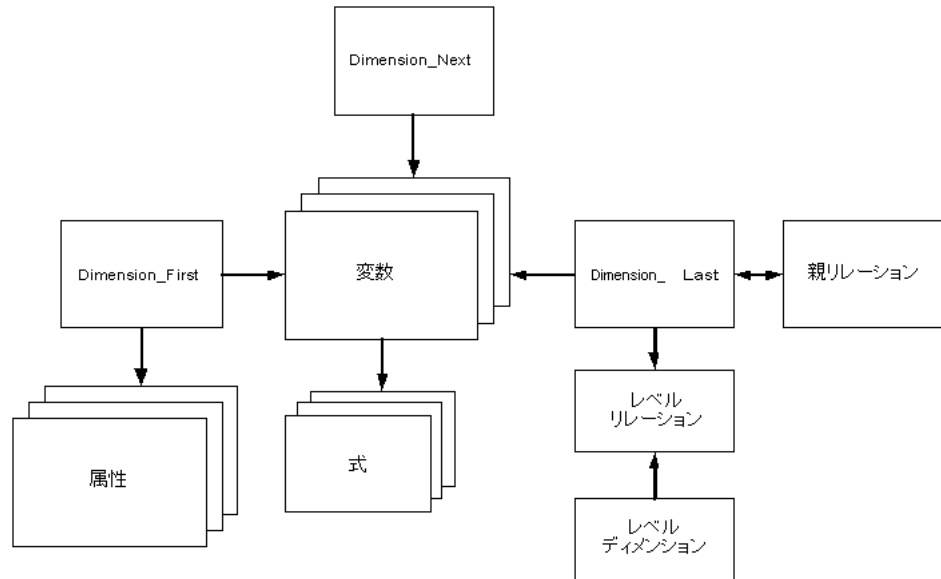
この図では、2001 年 1 月（表示）および 2001 年 2 月（非表示）における様々な地域の様々な製品の売上が比較されています。このデータ表示は、特定の市場における売上の少ない製品の識別に使用できます。次の図では、Rome（表示）および Tokyo（非表示）における 4 か月間の様々な製品の売上が表示されています。データの表示は傾向分析の基礎となります。



データベース・スタンダード・フォームのアナリティック・ワークスペース

スタンダード・フォームのアナリティック・ワークスペース内のディメンション、変数、式およびリレーション・オブジェクトが、多次元モデルの実装にどのように使用されるのかを [図 2-3](#) に示します。論理キューブは、同一のディメンションを持つメジャーによって構成されます。すべてのディメンションは属性を持ち、すべての階層ディメンションはレベル・リレーションおよびセルフ・リレーションを持ちます。明確にするため、これらのオブジェクトは図に 1 つのみ表示しています。変数および式は任意の数のディメンションを持つことができます。ここでは 3 つのディメンションが表示されています。

図 2-3 スタンダード・フォームのアナリティック・ワークスペースの図



アナリティック・ワークスペースのディメンション

アナリティック・ワークスペース内のディメンションは、高度に最適化された、キー表として機能する、値の1次元索引です。変数、リレーションおよび式（格納済の方程式）がディメンションを持つことができます。

ディメンションは、データ分析にとって重要で本質的な特性をいくつか備えています。

- 参照整合性。各ディメンション・メンバーは一意で、NA（つまり NULL）は許可されません。メジャーに3つのディメンションがある場合、そのメジャーの各データ値は各ディメンションのメンバーで修飾されている必要があります。また、ディメンション・メンバーの各組合せは、NAの場合でも値を持ちます。
- 一貫性。ディメンションは別々のコンテナとして保持され、メジャーによって共有されます。同じディメンションを持つメジャーなら、一緒に操作することが簡単にできます。たとえば、sales メジャーおよび expense メジャーが time と line でディメンション化されている場合、 $\text{profit} = \text{sales} - \text{expense}$ などの方程式を作成することができます。
- メンバーの保存順序。各ディメンションには、デフォルトのステータス（すべてのメンバーの格納順リスト）があります。デフォルトのステータス・リストは、メンバーの追加、削除または移動によって意図的に変更された場合以外は常に同じです。セッション中、ユーザーはステータス・リストの選択内容および順序を変更できます。変更したリストを現行のステータス・リストと呼びます。現行のステータス・リストは、メンバーの追加、削除または順序変更によって意図的に変更されないかぎり同じ状態です。

ディメンション・メンバーの順序には一貫性があるため、メンバーを相対的に選択することができます。たとえば、次のファンクション・コールでは、現行のステータス・リストの現在選択されているすべての期間の売上げ値と、前の期間の売上を比較します。

```
lagdif(sales, 1, time)
```

- 高度な非正規化。ディメンションは通常、すべての階層のすべてのレベルのメンバーを持ちます。このタイプのディメンションを、**埋込み合計**ディメンションと呼ぶ場合もあります。

単純なディメンションの他にも、スタンダード・フォームのアナリティック・ワークスペースで使用されるディメンションには、コンポジット・ディメンションおよび CONCAT ディメンションなど、特殊なタイプのディメンションがいくつかあります。これらのディメンションは、このマニュアルの後の章で説明します。

スタンダード・フォームのアナリティック・ワークスペースにおけるディメンションの使用

アナリティック・ワークスペースでは、異なるレベルのデータの集計、アロケーションおよびナビゲーションの操作ができるように、データ・ディメンションは階層的な構造になっています。ただし、すべての階層のすべてのレベルにおけるディメンション・メンバーはすべて、単一のデータ・ディメンション・コンテナに格納されます。たとえば、月、四半期および年はすべて、Time の単一のディメンションに格納されます。ディメンション・メンバー間の階層関係は、親子レレーションによって定義されます (2-10 ページの「[アナリティック・ワークスペースのリレーション](#)」を参照)。

現実にはすべてのデータが階層的ではなく、レベルを持たないデータ・ディメンションを作成することができます。勘定科目ディメンションはそのようなディメンションの 1 つで、このメンバー間の関係には、複数レベル階層ではなくモデルが必要です。アナリティック・ワークスペースで使用可能な拡張データ・モデリング・サブシステムによって、単独で解決できたり、集計メソッドとの併用で解決できる、単純なモデルと複雑なモデルの両方を作成することができます。

1 次元の索引として、ディメンション・コンテナには、メジャーのディメンション化に加え、アナリティック・ワークスペースで多くの使用方法があります。スタンダード・フォームのアナリティック・ワークスペースはディメンションを使用して、階層、レベルおよび論理キューブを構成するディメンションのリストなど、様々なタイプのメタデータを格納します。

アナリティック・ワークスペースのリレーション

リレーションは、データ型以外は変数と同一です。変数は、DECIMAL や TEXT などの一般的なデータ型を持ち、リレーションは、そのデータ型としてディメンションを持ちます。たとえば、PRODUCT のデータ型を持つリレーションは、PRODUCT ディメンションのメンバーである値しか受け付けず、その他の値を格納しようとしてもエラーが発生します。ディメンション・メンバーは、リレーションが受け付けることができる値の制限リストを提供します。リレーション・コンテナは、多次元であり得る点を除いて、外部キー列と似ています。

スタンダード・フォームのアナリティック・ワークスペースでは、親子レレーションおよびレベル・リレーションの2つがデータ・ディメンションの階層内容をサポートしています。

親子レレーションは、ディメンション階層の各メンバーの親を識別するセルフ・リレーションです。このリレーションは、ディメンションの階層構造を定義します。

レベル・リレーションは、ディメンション階層の各メンバーのレベルを識別します。これは、レベルに基づいたディメンション・メンバーの選択およびソートに使用されます。

アナリティック・ワークスペースの変数

変数は、データ値の表、つまり、特定のデータ型を持ち、ディメンションの特定のリストによって索引付けられた配列です。ディメンション自身は変数で格納されません。

ディメンション・メンバーの各組合せは、セルに値が存在するかどうかにかかわらず、データ・セルを定義します。したがって、データの不在を意図的に分析に含めたり、分析から除外することができます。たとえば、特定の製品が特定の日付以前で入手できなかった場合、分析者は、その前の期間のNAを除外できます。ただし、製品は入手できたが特定の市場で売らなかった場合には、分析者はNAを含めることができます。

同じディメンションを共有する複数の変数の間には、特別な物理的関係は存在しません。ただし、論理的な関係は存在します。変数は、異なるデータ型の可能性がある異なるデータを格納していても、同一のコンテナであるためです。同一のディメンションを持つ変数によって論理キューブは構成されます。

Time ディメンションに新しい期間を追加するなど、ディメンションを変更する場合、Time によってディメンション化されているすべての変数は、他の変数にこれらのデータがない場合でも、自動的に新しい期間を含むよう変更されます。ディメンションを共有する複数の変数を、集計、アロケーション、モデリングおよび数値計算など様々な方法と一緒に操作することもできます。アナリティック・ワークスペースでは、このタイプの計算は簡単で高速に行われますが、リレーショナル・スキーマでは同等の単一行計算が非常に困難となる場合があります。

変数を使用したメジャーの格納

アナリティック・ワークスペースでは、ファクトは通常、数値データ型で変数に格納されます。データの各型は、自身の変数内に格納されます。これによって、売上データおよび費用データが同じディメンションおよび同じデータ型を持つ一方、それぞれが2つの別個の変数に格納されます。コンテナは同一ですが、内容は一意となります。

アナリティック・ワークスペースは、サマリー・データの作成、格納およびメンテナンスに関して、マテリアライズド・ビューにかわる重要なものです。非常に高度な集計システムによって、多くの広範な集計方法に加え、モデリングがサポートされます。さらに、ファイブ・グレイン集計規則によって、1つのメジャー内のどのデータを事前集計し、同じメジャー内のどのデータを実行時に計算するのかを的確に決めることができます。

事前集計されたデータはベース・レベル・データと同じコンテナに稠密に格納されます。既知の優れた方法に従って集計規則を定義した場合、データをその場で集計することによるパ

パフォーマンスへの影響は無視できます。結果セットに必要な集計データが変数に格納されている場合、そのデータが単に取得されます。集計データがない場合、その場で計算されます。

変数を使用した属性の格納

メジャー同様、属性は変数に格納されます。ただし、属性とメジャーには大きな違いがあります。多くの場合、属性は多次元ですが、データ・ディメンションであるのは1つのディメンションのみです。データ・ディメンション階層を示す階層ディメンション、および複数言語をサポートする言語ディメンションがその他のディメンションの典型です。

属性は、ディメンション階層のレベルにかかわらず、各ディメンション・メンバーに関する補足情報を提供します。たとえば、**Time** ディメンションが、説明的な名前、最終日および期間用にそれぞれ1つずつ、合計3つの属性変数を持つ場合があります。これらの属性では、説明的な名前が **October 2002**、終了日が **31-OCT-02** および期間が **31** の **Time** メンバー **OCT-02** などが提供されます。**Time** ディメンションのその他の日、月、四半期および年はすべて、これら3つの属性変数に格納されているものと同様の情報を持ちます。

アナリティック・ワークスペースの式

式とは、格納済の方程式のことです。OLAP DML の任意のファンクションへのコールまたは任意のカスタム・プログラムへのコールを式に格納できます。このように、アナリティック・ワークスペースの式はリレーショナル・ビューと似ています。

スタンダード・フォームのアナリティック・ワークスペースでは、式オブジェクトの使用法の1つは、集計規則とデータを保持する変数との間のインタフェースを提供することです。メジャーの名前は常に式の名前であって、基になる変数の名前ではありません。変数には格納済データ（ベース・レベルのデータおよび事前計算済の集計）しか格納されないのに対して、式は、格納済データおよびその場で計算されたデータの両方を格納し、完全に解決済のメジャーを戻します。これによって、特定のメジャーに対するすべての問合せが、データが計算されたものか単に取得されたものかにかかわらず、アナリティック・ワークスペースの同じリレーショナル・ビューの同じ列に対して書き込まれます。その列が式から取得したデータを示します。

式は、率、差、移動合計および移動平均などのその他の結果の即時計算に使用することもできます。

サンプル・スキーマ

このマニュアルでは、例として Global スキーマを使用します。この章ではこのスキーマについて説明し、多次元オブジェクトにこのスキーマをどのようにマップするのかを示します。この章では、次の項目について説明します。

- 事例のシナリオ
- Global スター・スキーマ

事例のシナリオ

仮に、Global Computing Company が 1990 年に設立されたものとします。Global Computing は、コンピュータのハードウェアおよびソフトウェアのコンポーネントを世界規模で顧客に供給しています。営業部門では、見積値を満たしていません。結果として、この部門では、成功を収める営業戦略の開発に挑戦してきました。

Global Computing は、きわめて競争の激しい市場の中で経営されています。競争相手は多数あり、顧客は価格に特に敏感で、利益幅は少なくなる傾向にあります。収益を高めて発展するために、Global Computing は最も利益になる製品の売上を増やす必要があります。

Global Computing の現在のビジネスにおける次の様々な要素が、売上および収益における低下を示しています。

- 伝統的に、Global Computing では、第 3 四半期（7 月から 9 月）に売上が低下しています。しかし、最近では、他の四半期の売上も予測を下回っています。この会社は急速な成長を経験しましたが、明確な理由もなく過去 2 年間では、第 1 四半期の売上がその前までの年と比較して低くなっています。
- Global は、最新の販売チャネルであるインターネットでは成功を収めています。このチャネルにおける売上は伸びていますが、全体の収益は減少しています。
- パーソナル・コンピュータ（以前の Global Computing のほとんどの収益の源泉）の利益が急速に減少していることが最も重要な要因かもしれません。

Global Computing は、これらの各要素がビジネスにどのように影響しているのかを理解する必要があります。

現在のレポート作成は IT 部門によって行われ、月単位の標準的なレポートが作成されています。非定形のレポートはすべて必要に応じて処理されますが、限られた IT スタッフの時間に制約されています。営業部門では、レポートの要求に対する応答の遅れに関して不満が広がっています。IT 部門でも、方針を頻繁に変更し、より詳細な情報を求める分析者に対する不満が蓄積しています。

営業部門では、何が売れているのか、誰が購入しているのか、どのように購入しているのかといったタイムリな情報の欠如に苦闘しています。情報主任との打合せでは、営業部長が「情報は取得した時点でもう役に立たない。情報を取得できるのは月末で、それには仕事で必要な詳細な情報が含まれていない」と報告しています。

要件のレポート

必要なことに関してより詳しく尋ねられると、営業部長は次の要件を特定しています。

- 特定の顧客、地域およびセグメントの売上データの傾向。
- 現地販売員に対する情報および一定の情報分析能力の提供。販売員は世界中に分散しているため、Web インタフェースが好ましい。
- 週および月単位の通信販売、電話販売および電子メール販売に関する詳細（過去の期間との比較も含める）。情報を使用して、各チャネルでいつ、どうやって何が売れているのかを確認できる必要がある。
- 各売上のドル貢献を知るための、製品に関する利益情報。
- 売上、単位および利益の、前の期間および 1 年前の期間との変化率。
- 非定型のグルーピングによるデータ分析の実行。

情報主任はチーム内でこれらの要件について話し合い、生産受注システムに対する標準的なレポート・ソリューションは、必要とされる分析能力を提供するだけの柔軟性がないという結論に達しました。ビジネス分析のレポート要件があまりにも多様なので、要求に対する期待される応答時間に加えて開発の見積りコストが、このソリューションを容認できないものとしています。

情報主任のチームは、分析をサポートするためにアナリティック・ワークスペースの使用を推奨しています。チームでは、営業部門の IT グループが IT 企業と連携し、彼らの要件を満たす情報分析用のアナリティック・ワークスペースを構築することを提案しました。

ビジネスの目標

開発チームは、このプロジェクトで満たす必要のある高水準のビジネス目標を特定しました。

- Global Computing の戦略上の目標として、より高い利益の製品の売上および全体の売上高を伸ばすことによって、会社の収益を増加させます。
- 営業部門の目標は次のとおりです。

- 業界の傾向を分析し、特定の市場セグメントを標的にする。
- 販売チャネルを分析し、収益を向上させる。
- 製品の傾向を識別し、適切なチャネルを開拓する戦略を作成する。

情報の要件

ビジネスの目標を確立したら、それらの目標の達成に役立つ情報のタイプを決定できます。エンド・ユーザーがアナリティック・ワークスペースのデータをどのように調べるかを理解するために、広範囲のヒアリングを実施することが重要です。キーとなるエンド・ユーザーのヒアリングから、ビジネスに対する調査方法および回答が欲しいビジネス分析の問合せのタイプを決定できます。

ビジネス分析の問合せ

Global Computing の営業部長、販売員および市場分析者からのヒアリングによって、次のビジネス分析の問合せが明らかになりました。

- どの製品が利益が高いか？
- 顧客は誰か、彼らは何をどのように購入するのか？
- どのアカウントが最も収益性が高いか？
- 各流通チャネルの業績は？
- ビジネスに季節的な分散はまだ存在するか？

次に、これらのビジネス分析の各問合せを詳細に調査します。

どの製品が利益が高いか？

このビジネス分析の問合せは次の問合せから構成されます。

- 任意の月、四半期、年または任意の流通チャネルにおける任意の項目、製品ファミリー、製品クラスの総売上上の比率は？この売上比率が1年前とどのように異なるか？
- 任意の特定月における任意の項目の各単位の単位価格、単位原価および単位当たりの利益は？任意の月における任意の項目の価格、原価および利益の傾向は？
- 任意の流通チャネルおよび任意の地域または市場セグメントにおける、任意の月、四半期、年の最も利益の高かった項目は何か？収益性は前の期間からどのように変化したか？前の期間からの収益性の変化率は？
- 前の期間から収益性が最も変化した項目は何か？
- 任意の流通チャネルおよび任意の地域または市場セグメントにおける、任意の月、四半期、年の総収益に最も貢献した項目は何か？
- 任意の特定月の単位当たりの利益が最も高い項目は何か？

- 要約すると、傾向は何か？

顧客は誰か、彼らは何をどのように購入するのか？

このビジネス分析の問合せは次の問合せから構成されます。

- 任意の月、四半期または年における任意の項目、製品ファミリー、製品クラスの売上は？
- 任意の流通チャネル、地域または市場セグメントにおける任意の項目、製品ファミリー、製品クラスの売上は？
- 売上は前の期間からどのように変化したか？前の期間からの売上の変化率は？
- 売上は1年前からどのように変化したか？1年前からの売上の変化率は？
- 要約すると、傾向は何か？

どのアカウントが最も収益性が高いか？

このビジネス分析の問合せは次の問合せから構成されます。

- 任意の項目、製品ファミリー、製品クラスの、任意の流通チャネルにおける任意の月、四半期、年の最も利益の高いアカウントは何か？
- 任意の流通チャネルおよび任意の製品の、任意の月、四半期、年のアカウントの、売上および総利益（粗利益）は？
- アカウントの収益性を前の期間とどのように比較するか？
- どのアカウントが前の期間と比較して最も売上が伸びたか？
- 前の期間からの売上の割合変化は？収益性の変化率が、売上の変化率と同じ比率で増加したか？
- 要約すると、傾向は何か？

各流通チャネルの業績は？

このビジネス分析の問合せは次の問合せから構成されます。

- 任意の項目、製品ファミリー、製品クラスまたは任意の地域、市場セグメントの各流通チャネルの総売上に対する売上比率は？
- 各流通チャネル（直販、カタログ販売およびインターネット）の収益性は？
- 最新の流通チャネルのインターネットがカタログ販売と共食いついていないか？顧客が注文方法を切り替えているだけなのか、インターネット流通チャネルがさらなる顧客を獲得しているのか？
- 要約すると、傾向は何か？

ビジネスに季節的な分散はまだ存在するか？

このビジネス分析の問合せは次の問合せから構成されます。

- 特定の項目または製品ファミリの季節的な売上パターンを識別できるか？
- 季節的な売上パターンは地域によってどのように異なるか？
- 季節的な売上パターンは市場セグメントによってどのように異なるか？
- 季節的な売上パターンは昨年と比較して違いはあるか？

情報の要件のまとめ

エンド・ユーザーが行いたい分析のタイプを調査することによって、分析のキーとなる次の要件を識別できます。

- Global Computing には、収益性の分析に対する強い要求がある。会社は、製品、アカウント、市場セグメントおよび流通チャネルごとの収益性を解釈する必要がある。また、収益性の傾向についても解釈する必要がある。
- Global Computing は、売上が時期によってどのように異なるのかについて解釈する必要がある。会社は、製品、地域、市場セグメントおよび流通チャネルごとのこれらの季節的な傾向について解釈する必要がある。
- Global Computing には、非定形の売上分析に対する強い要求がある。分析者は、どの製品が誰にいつ売れているのか、顧客はどのようにこれらの製品を購入しているのかを識別する必要がある。
- 傾向分析力が Global Computing にとって重要である。

必要とされるビジネス・ファクトの識別

分析のキーとなる要件によって、Global Computing における分析要件のサポートに必要なビジネス・ファクトが明らかになります。

次のファクトは、時間、製品、顧客への発送、市場セグメントおよび流通チャネルによって整理されます。

売上

単位

前の期間からの売上の変化

前の期間からの売上の変化率

前の年からの売上の変化

前の年からの売上の変化率

製品シェア

チャネル・シェア

市場シェア

総費用

総利益

前の期間からの総利益の変化
前の期間からの総利益の変化率
製品の総売上に対する利益の割合
前の期間からの販売単位の変化
前の期間からの販売単位の変化率
前の年からの販売単位の変化
前の年からの販売単位の変化率

次のファクトは項目および月によって整理されます。

単位価格
単位原価
単位当たりの利益

Global Computing のための論理データ・モデルの設計

Global Computing における分析要件をサポートするビジネス・ファクトを 3-2 ページの「[ビジネスの目標](#)」に示します。次に、論理データ・モデル内のディメンション、レベルおよび属性を識別します。各ディメンション内の関係も識別します。結果としてできるデータ・モデルは、Global スター・スキーマ、OLAP カタログ・メタデータおよびアナリティック・ワークスペースの設計に使用します。

ディメンションの識別

4 つのディメンションをデータベース内のファクトの構成に使用します。

- Product は、製品によってデータがどのように異なるかを示します。
- Customer は、顧客や地域によってデータがどのように異なるかを示します。
- Channel は、各流通チャネルによってデータがどのように異なるかを示します。
- Time は、時間によってデータがどのように異なるかを示します。

レベルの識別

ディメンションを識別したところで、各ディメンション内の集計のレベルを識別できます。Global Computing での分析要件によって、次のことが明らかになっています。

- 流通チャネルには、Sales、Catalog および Internet の 3 つがあります。これらの 3 つの値は、データ・ウェアハウスにおける細目の最低レベルであり、Channel レベル内でグループ化されます。レベルは、集計の最高レベルから細目の最低レベルの順に、All Channels および Channel となります。
- Global では、顧客への発送取扱いに沿って、および市場セグメントごとに、顧客および地域の分析を行います。それぞれの場合で、データ・モデルにおける細目の最低レベルは Ship To 位置です。

- 顧客への発送取扱いに沿って分析する場合、集計のレベルは（高い方から低い方の順に）、All Customers、Region、Warehouse および Ship To となります。
- 市場セグメントによって分析する場合、集計のレベルは（高い方から低い方の順に）、Total Market、Market Segment、Account および Ship To となります。
- Product ディメンションは（高い方から低い方の順に）、Total、Class、Family および Item の 4 つのレベルを持ちます。
- Time ディメンションは（高い方から低い方の順に）、Year、Quarter および Month の 3 つのレベルを持ちます。

Channel、Customer および Product ディメンション内には、集計の最高レベルとして、Total または All レベルを追加しました。この最高レベルを追加することによって、アプリケーション・ユーザーはデータを柔軟に分析できるようになります。

階層の識別

各ディメンション内のレベルを構成する階層を識別します。階層を識別するには、集計の正しい順序で、および分析の識別したタイプをサポートする方法で、レベルをグループ化します。

Global Computing では、Channel、Product および Time の各ディメンションについて必要な階層は 1 つのみです。ただし、Customer ディメンションについては、Global Computing では 2 つの階層が必要となります。Customer ディメンションにおける分析は、地域または市場セグメントのいずれかによるものになります。したがって、Shipments および Market Segment の 2 つの階層にレベルを構成します。

ストアド・メジャーの識別

3-5 ページの「[必要とされるビジネス・ファクトの識別](#)」では、Global Computing の分析要件のサポートに必要な 21 のビジネス・ファクトを示しています。このうち、トランザクション・データベースから取得される必要があるファクトは次の 3 つのみです。

- 単位
- 単位価格
- 単位原価

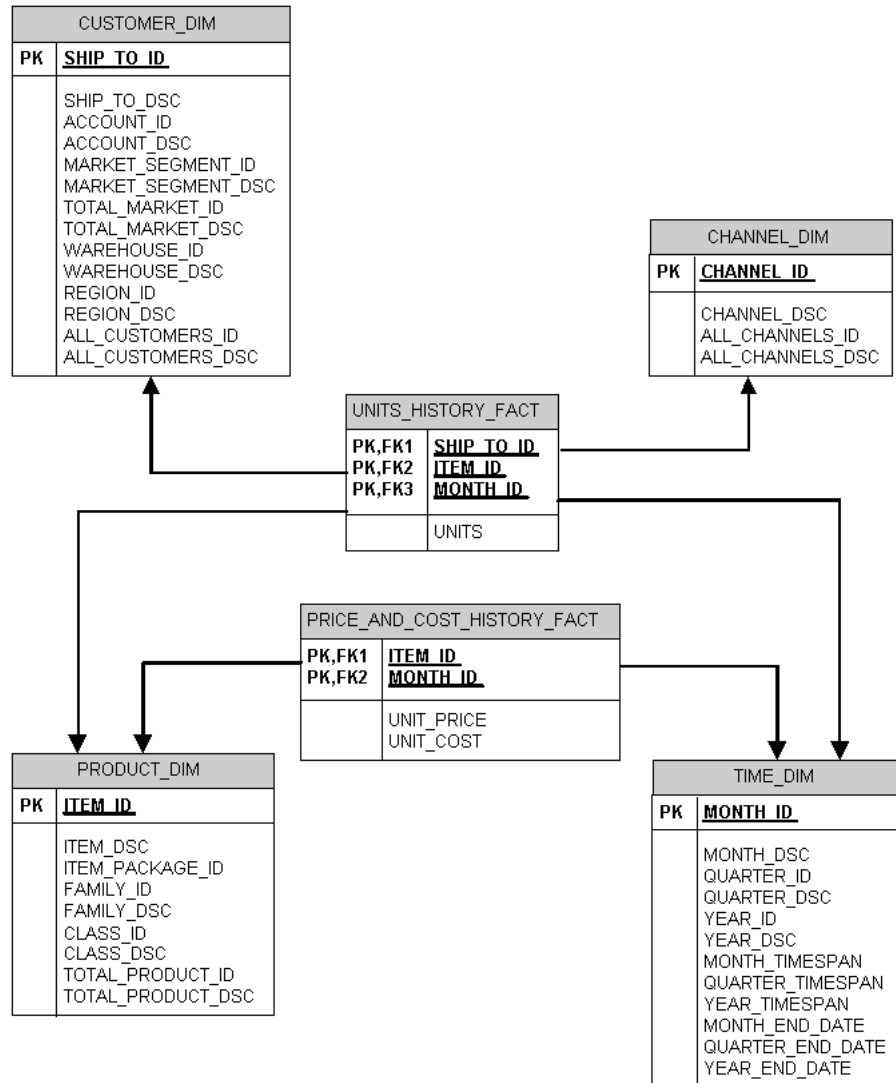
その他のすべてのファクトは、これらの基本ファクトから導出できます。導出されたファクトは、オンデマンドでアナリティック・ワークスペース内で計算できます。これらの導出ファクトの一部が頻繁に使用され、その計算がシステムに大きな負荷を与えていることが経験上明らかな場合、これらのファクトは、データのメンテナンス処理として、アナリティック・ワークスペース内で計算および格納できます。

Global スター・スキーマ

Global スキーマは、2 つのファクト表と 4 つのディメンション表から構成されます。ディメンション表は、各レベル列にサロゲート・キー（数値）を使用して、ディメンション・メンバーがすべてのレベルで一意的であることを保証します。たとえば地理ディメンションは、City レベルでの New York と State レベルでの New York などのように、異なるレベルで同じ値を容易に持つ可能性があります。アナリティック・ワークスペースでは、すべてのレベルのディメンション・メンバーが単一のディメンションにフェッチされ、一意性を保証するために追加の手順を行わないかぎり、重複した値がお互いを上書きします。

[図 3-1](#) に、表間の関係を示します。さらに、Global スキーマには、図では省略されていますが、UNITS_HISTORY_FACT 表と同じ論理位置を占有する UNITS_UPDATE_FACT 表が含まれます。

図 3-1 Global スキーマの図



ディメンション表：TIME_DIM

TIME_DIM 表は、3つのレベルを持つ時間ディメンションを定義します。各レベルは、サロゲート・キー（数値）、テキストの説明、終了日（期間の最後の日）および期間（期間の日

数) の 4 つの列によってサポートされています。これは、時間ディメンションの定義に必要な最も基本的な情報です。

サロゲート・キーは、表のコンテキスト外では意味を持たない人工的な値です。これによって、異なるレベルで同じ値が重複しないことが保証され、リレーショナル表およびアナリティック・ワークスペースの両方で最高の処理速度を提供します。説明列ではこれらの数値識別子に対する意味が提供されます。

終了日列および期間列は、次のような時系列分析をサポートします。

- 前の期間からの変更
- 1 年前からの変更
- 年度累計
- 時間の範囲

1998 から 2004 まで 7 つの年が定義されており、1998 から 2003 の初頭までデータが提供されています。最後の年が予測に使用できます。

標準的な階層におけるベース・レベルからトップ・レベルまでのロールアップ順序は次のとおりです。

MONTH -> QUARTER -> YEAR

表 3-1 に、TIME_DIM 表の列を示します。

表 3-1 TIME_DIM の列の説明

列	データ型	ロール	一意値	サンプル値
MONTH_ID	NUMBER	主キー サロゲート・キー	78	34
MONTH_DSC	VARCHAR2	属性		Apr-99
QUARTER_ID	NUMBER	サロゲート・キー	26	10
QUARTER_DSC	VARCHAR2	属性		Q2-99
YEAR_ID	NUMBER	サロゲート・キー	7	2
YEAR_DSC	VARCHAR2	属性		1999
MONTH_TIME_SPAN	NUMBER	属性		30
QUARTER_TIMESPAN	NUMBER	属性		91
YEAR_TIMESPAN	NUMBER	属性		365
MONTH_END_DATE	DATE	属性		30-Apr-1999
QUARTER_END_DATE	DATE	属性		30-Jun-1999

表 3-1 TIME_DIM の列の説明

列	データ型	ロール	一意値	サンプル値
YEAR_END_DATE	DATE	属性		31-Dec-1999

ディメンション表 : CUSTOMER_DIM

CUSTOMER_DIM 表は、2つの階層の定義に使用される7つのレベルを定義します。各レベルはサロゲート・キー（数値）およびテキストの説明を持ち、これは通常のディメンション（つまり、時間ディメンションではない）を定義するための最も基本的な情報です。SHIP_TO は主キーで、その値は、Customer の両方の階層のベース・レベル・メンバーとなります。

Market 階層におけるベース・レベルからトップ・レベルまでのロールアップ順序は次のとおりです。

SHIP_TO -> ACCOUNT -> MARKET_SEGMENT -> TOTAL_MARKET

Customer 階層におけるロールアップ順序は次のとおりです。

SHIP_TO -> WAREHOUSE -> REGION-> ALL_CUSTOMERS

表 3-2 に、CUSTOMER_DIM 表の列を示します。

表 3-2 CUSTOMER_DIM の列の説明

列	データ型	ロール	階層	一意値	サンプル値
SHIP_TO_ID	NUMBER	主キー サロゲート・キー	両方	61	89
SHIP_TO_DSC	VARCHAR2	属性			Monolith Motor Co. Knoxville
ACCOUNT_ID	NUMBER	サロゲート・キー	Market	24	36
ACCOUNT_DSC	VARCHAR2	属性			Monolith Motor Company
MARKET_SEGMENT_ID	NUMBER	サロゲート・キー	Market	5	5
MARKET_SEGMENT_DSC	VARCHAR2	属性			Manufacturing
TOTAL_MARKET_ID	NUMBER	サロゲート・キー	Market	1	7
TOTAL_MARKET_DSC	VARCHAR2	属性			Total Market
WAREHOUSE_ID	NUMBER	サロゲート・キー	Customers	11	21
WAREHOUSE_DSC	VARCHAR2	属性			United States

表 3-2 CUSTOMER_DIM の列の説明

列	データ型	ロール	階層	一意値	サンプル値
REGION_ID	NUMBER	サロゲート・キー	Customers	3	10
REGION_DSC	VARCHAR2	属性			North America
ALL_CUSTOMERS_ID	NUMBER	サロゲート・キー	Customers	1	1
ALL_CUSTOMERS_DSC	VARCHAR2	属性	Customers		All Customers

ディメンション表 : PRODUCT_DIM

PRODUCT_DIM 表は、4 つのレベルを持つ Product ディメンションを定義します。各レベルは、サロゲート・キー（数値）および説明テキストを持ちます。ITEM_ID は主キーで、その値は、Product ディメンションのベース・レベル・メンバーとなります。

Product 階層におけるベース・レベルからトップ・レベルまでのロールアップ順序は次のとおりです。

ITEM -> FAMILY -> CLASS -> TOTAL_PRODUCT

表 3-3 に、PRODUCT_DIM 表の列を示します。

表 3-3 PRODUCT_DIM の列の説明

列	データ型	ロール	一意値	サンプル値
ITEM_ID	NUMBER	主キー サロゲート・キー	36	48
ITEM_DSC	VARCHAR2	属性		Keyboard Wrist Rest
ITEM_PACKAGE_ID	NUMBER	属性	4	Laptop Value Pack
FAMILY_ID	NUMBER	サロゲート・キー	9	7
FAMILY_DSC	VARCHAR2	属性		Accessories
CLASS_ID	NUMBER	サロゲート・キー	2	3
CLASS_DSC	VARCHAR2	属性		Software/Other
TOTAL_PRODUCT_ID	NUMBER	サロゲート・キー	1	1
TOTAL_PRODUCT_DSC	VARCHAR2	属性		Total Product

ディメンション表 : CHANNEL_DIM

CHANNEL_DIM 表には 4 つの列が格納されます。CHANNEL_ID は主キーで、その値は、Channel ディメンションのベース・レベル・メンバーとなります。ALL_CHANNELS_ID は、

チャンネルのすべてを表す単一の値を定義します。OLAP カタログでは、これらの 2 つの列が単一の Channel 階層の 2 つのレベルを定義します。ベース・レベルからトップ・レベルまでのロールアップ順序は次のとおりです。

CHANNEL -> ALL_CHANNELS

残りの列の CHANNEL_DSC および ALL_CHANNELS_DSC は、サロゲート・キーに意味を与えるテキストの説明を提供します。

表 3-4 に、CHANNEL_DIM 表の列を示します。

表 3-4 CHANNEL_DIM の列の説明

列	データ型	ロール	一意値	サンプル値
CHANNEL_ID	NUMBER	主キー サロゲート・キー	3	4
CHANNEL_DSC	VARCHAR2	属性		Internet
ALL_CHANNELS_ID	NUMBER	サロゲート・キー	1	1
ALL_CHANNELS_DSC	VARCHAR2	属性		All Channels

ファクト表 : UNITS_HISTORY_FACT および UNITS_UPDATE_FACT

UNITS_HISTORY_FACT および UNITS_UPDATE_FACT 表は 4 つの外部キー列を格納し、それらが一緒になって複数列の主キーを構成します。外部キーは、4 つのディメンション表の主キーに関連付けられます。

UNITS_HISTORY_FACT では、Product、Customer および Channel の各外部キー値が少なくとも 1 度は使用され、65 の期間が使用されます。表は、考え得る 513,864 の一意キーの組合せのうち 169,487 行を格納します。

UNITS_UPDATE_FACT は、月 91 (Jun-03) のデータを追加します。Product、Customer および Channel の各外部キー値は少なくとも 1 度は使用されます。表は、考え得る 6,588 のキーの組合せのうち、3,459 行を格納します。

表 3-5 に、両方の表の列を示します。

表 3-5 UNITS_HISTORY_FACT および UNITS_UPDATE_FACT の列の説明

列	データ型	ロール	説明
CHANNEL_ID	NUMBER	キー	CHANNEL_DIM に関連付けられる
ITEM_ID	NUMBER	キー	PRODUCT_DIM に関連付けられる
SHIP_TO_ID	NUMBER	キー	CUSTOMER_DIM に関連付けられる
MONTH_ID	NUMBER	キー	TIME_DIM に関連付けられる

表 3-5 UNITS_HISTORY_FACT および UNITS_UPDATE_FACT の列の説明

列	データ型	ロール	説明
UNITS	NUMBER	ファクト	販売単位数

ファクト表 : PRICE_AND_COST_HISTORY_FACT および PRICE_AND_COST_UPDATE_FACT

PRICE_AND_COST_HISTORY_FACT および PRICE_AND_COST_UPDATE_FACT 表は、2 つの外部キー列（一緒に複数列の主キーを構成する）および 2 つのファクト列を格納します。

PRICE_AND_COST_HISTORY_FACT では、65 の月のすべての製品にデータが提供されます。

PRICE_AND_COST_UPDATE_FACT は、すべての製品の月 91（Jun-03）のデータを追加します。

表 3-6 に、両方の表の列を示します。

表 3-6 PRICE_AND_COST_HISTORY_FACT および PRICE_AND_COST_UPDATE_FACT の列の説明

列	データ型	ロール	説明
ITEM_ID	NUMBER	キー	PRODUCT_DIM に関連付けられる
MONTH_ID	NUMBER	キー	TIME_DIM に関連付けられる
UNIT_PRICE	NUMBER	ファクト	単位価格のリスト
UNIT_COST	NUMBER	ファクト	単位原価

アナリティック・ワークスペースへの Global スキーマのマッピング

OLAP カタログは、リレーショナル表の列をアナリティック・ワークスペースの多次元オブジェクトにマッピングするためのインタフェースを提供します。次の表では、PRICE_AND_COST_HISTORY_FACT ファクト表およびその関連付けられたディメンション表の PRODUCT_DIM および TIME_DIM のマッピングを識別します。これらの表は、PRICE_CUBE キューブのソースです。

これらの表に示されているアナリティック・ワークスペース・オブジェクトは第 6 章「アナリティック・ワークスペースの作成」で作成します。詳細は、第 8 章「スタンダード・フォームのアナリティック・ワークスペースの詳細」を参照してください。

Global Product ディメンションのマッピング

表 3-7 に、PRODUCT_DIM ディメンション表の列がどのようにしてワークスペース・オブジェクトにマップされ、**埋込み合計**の PRODUCT ディメンションを提供するのかが示します。

PRODUCT_DIM は、ITEM、FAMILY、CLASS および TOTAL_PRODUCT の 4 つのレベルを持つ単一のディメンション階層 PRODUCT_ROLLUP をサポートします。説明列は、Long Description および Short Description の両方にマップされますが、アナリティック・ワークスペースではこの重複は必要ありません。

表 3-7 Global Product ディメンションのマッピング

PRODUCT_DIM 表の列	OLAP カタログの論理オブジェクト	GLOBAL アナリティック・ワークスペースのオブジェクト
ITEM_ID	PRODUCT_ROLLUP 階層、ITEM レベル	PRODUCT ディメンション
FAMILY_ID	PRODUCT_ROLLUP 階層、FAMILY レベル	PRODUCT_PARENTREL 親子関係
CLASS_ID	PRODUCT_ROLLUP 階層、CLASS レベル	PRODUCT_LEVELLIST レベル・ディメンション (ITEM、FAMILY、CLASS、TOTAL_PRODUCT)
TOTAL_PRODUCT_ID	PRODUCT_ROLLUP 階層、TOTAL_PRODUCT レベル	PRODUCT_LEVELREL レベル・関係 PRODUCT_HIERLIST 階層ディメンション (PRODUCT_ROLLUP)
ITEM_PACKAGE	PACKAGE 属性	PRODUCT_PACKAGE 変数

表 3-7 Global Product ディメンションのマッピング

PRODUCT_DIM 表の列	OLAP カタログの 論理オブジェクト	GLOBAL アナリティック・ ワークスペースのオブジェクト
ITEM_DSC	ITEM の Long Description 属性 ITEM の Short Description 属性	PRODUCT_LONG_DESCRIPTION 変数 PRODUCT_SHORT_DESCRIPTION 変数
FAMILY_DSC	FAMILY の Long Description 属性 FAMILY の Short Description 属性	
CLASS_DSC	CLASS の Long Description 属性 CLASS の Short Description 属性	
TOTAL_PRODUCT_DSC	CLASS の Long Description 属性 CLASS の Short Description 属性	

Global Time ディメンションのマッピング

表 3-8 に、TIME_DIM ディメンション表の列がどのようにしてワークスペース・オブジェクトにマップされ、埋込み合計の TIME ディメンションを提供するのかが示します。

TIME_DIM は、Month、Quarter および Year の 3 つのレベルを持つ単一のディメンション階層 Calendar をサポートします。アナリティック・ワークスペースにおける時間ベースの分析のために、Time ディメンションは、ここで示すように End_Date および Time_Span 属性を持つ必要があります。説明列は、Long Description および Short Description の両方にマップされますが、アナリティック・ワークスペースではこの重複は必要ありません。

表 3-8 Global Time ディメンションのマッピング

TIME_DIM 表の列	OLAP カタログの 論理オブジェクト	GLOBAL アナリティック・ ワークスペースのオブジェクト
MONTH_ID	Calendar 階層、Month レベル	TIME ディメンション
QUARTER_ID	Calendar 階層、Quarter レベル	TIME_PARENTREL 親子リレーション
YEAR_ID	Calendar 階層、Year レベル	TIME_LEVELLIST レベル・ディメンション (Month、Quarter、Year) TIME_LEVELREL レベル・リレーション TIME_HIERLIST 階層ディメンション (Calendar)
MONTH_DSC	Month の Long Description 属性 Month の Short Description 属性	TIME_LONG_DESCRIPTION 変数 TIME_SHORT_DESCRIPTION 変数
QUARTER_DSC	Quarter の Long Description 属性 Quarter の Short Description 属性	
YEAR_DSC	Year の Long Description 属性 Year の Short Description 属性	
MONTH_TIMESPAN	Month の Time_Span 属性	TIME_TIME_SPAN 変数
QUARTER_TIMESPAN	Quarter の Time_Span 属性	
YEAR_TIMESPAN	Year の Time_Span 属性	
MONTH_END_DATE	Month の End_Date 属性	TIME_END_DATE 変数
QUARTER_END_DATE	Quarter の End_Date 属性	
YEAR_END_DATE	Year の End_Date 属性	

Global Price キューブのマッピング

表 3-9 に、PRICE_AND_COST_HISTORY_FACT ファクト表の列がどのようにしてワークスペース・オブジェクトにマップされ、UNIT_COST および UNIT_PRICE の 2 つのメジャーを持つ PRICE_CUBE キューブを提供するのかが示します。

集計演算子は OLAP カタログに定義され、キューブの初期 `aggmap` の基盤となります。`aggmap` によって集計の規則が提供されます。メジャー式は `aggmap` を使用して、メジャー変数にロードされるベース・レベルのデータを集計します。

変数のほとんどはスパースなため、キューブと関連付けられているコンポジット・ディメンションを必要とします。

表 3-9 Global Price キューブのマッピング

PRICE_AND_COST_HISTORY_FACT 表の列	OLAP カタログの論理オブジェクト	GLOBAL アナリティック・ワークスペースのオブジェクト
ITEM_ID	PRICE_CUBE キューブ SUM 集計演算子	PRICE_CUBE ディメンション (TIME、PRODUCT)
MONTH_ID		PRICE_CUBE_COMPOSITE ディメンション PRICE_CUBE_AGGMAP_AWCREATE DDEFAULT_1 aggmap
UNIT_PRICE	UNIT_PRICE メジャー	UNIT_PRICE 式 UNIT_PRICE_VARIABLE 変数
UNIT_COST	UNIT_COST メジャー	UNIT_COST 式 UNIT_COST_VARIABLE 変数

OLAP 用 Java アプリケーションの開発

この章では、OLAP アプリケーションを作成するために使用できる、豊富な機能を持つ開発環境および強力なツールについて説明します。この章では、次の項目について説明します。

- [分析 Java アプリケーションの作成](#)
- [BI Beans の概要](#)
- [OLAP API の理解](#)

アナリティック・ワークスペースへの SQL アクセスについては、[第7章「アナリティック・ワークスペースへの SQL アクセス」](#)を参照してください。

分析 Java アプリケーションの作成

Java はインターネットの言語です。Java を使用すると、アプリケーション開発者は、スタンドアロンの Java アプリケーション（Java の WebStart テクノロジーでブラウザから起動可能）を作成でき、サーブレットの JavaServer Pages (JSP) および Oracle User Interface XML (UIX) を介して、Oracle Database の生のデータにアクセスする HTML アプリケーションを作成できます。

Java の概要

Java は、増え続けるプロフェッショナルなソフトウェア開発者が最もよく使用するプログラミング言語です。Java は C 言語の短所を補うだけでなく、親しみやすい環境を提供するため、C または C++ を使用してきたプログラマは容易に Java へ移行しています。Sun Microsystems 社が開発した Java は、次の理由から、C++ および Visual Basic に代わるものとして、アプリケーション開発者が選択する言語になりつつあります。

- オブジェクト指向。Java を使用すると、アプリケーション開発者は、抽象的なプロセスではなく、データおよびそのデータを操作する方法に注目できます。つまり、プログラマは、必要なオブジェクトの作成に必要な手順ではなく、必要なオブジェクトを定義します。Java では、ほぼすべてのものがオブジェクトとして定義されます。

- プラットフォーム非依存。Java コンパイラによって、Java Virtual Machine (JVM) が実行時に変換するバイト・コードが作成されます。その結果、JVM がインストールされているすべての Windows、UNIX および Macintosh プラットフォームで同じソフトウェアを実行できます。すべての主なブラウザには JVM が組み込まれています。
- ネットワーク・ベース。Java はネットワーク上で動作するように設計されています。これによって、Java で作成されたプログラムは、ローカルのリソースと同様に、リモートのリソースを簡単に処理できます。
- 安全性が高い。Java コードはトラステッドまたはアントラステッドのいずれかであり、システム・リソースへのアクセスはこの特性によって判断されます。ローカル・コードはトラステッドであり、システム・リソースにフルアクセスできますが、ダウンロードしたリモート・コード（アプレット）はトラステッドではありません。Java のサンドボックス・セキュリティ・モデルでは、アントラステッド・コードに対して厳しく制限された環境が提供されます。

OLAP の Java ソリューション

OLAP アプリケーションを開発するには、Java プログラミング言語を使用できます。Java によって、インターネット上に簡単にデプロイできるプラットフォーム非依存のアプリケーションを作成できます。

OLAP API は、分析ビジネス・アプリケーションの多次元データにアクセスできる Java ベースのプログラミング・インタフェースです。OLAP API は OLAP カタログ・メタデータを使用して、スター・スキーマやスノーフレーク・スキーマのリレーショナル表に格納されているデータ、または OLAP API を使用可能なアナリティック・ワークスペースのビューに格納されているデータにアクセスします。

OLAP API の Java クラスによって、OLAP アプリケーションに必要なすべての機能が提供されます。たとえば、OLAP インスタンスへの接続、ユーザー証明書の認証、それらの証明書に付与された権限によって制御された RDBMS のデータへのアクセス、ビジネス分析のためのデータの選択および操作などです。

BI Beans では、これらの機能を JavaBeans として提供することにより、アプリケーションを簡単にデプロイできるようになります。さらに、BI Beans には、データをグラフ、クロス集計および表で表示するための JavaBeans も含まれています。

注意： Oracle JDeveloper および BI Beans は、Oracle RDBMS にはパッケージされていません。

Oracle の Java 開発環境

Oracle JDeveloper では、Java アプリケーションの開発用に統合開発環境（IDE）が提供されています。サード・パーティ製の Java IDE を効率的に使用することもできますが、Oracle Database と BI Beans ウィザードを完全に統合できるのは Oracle JDeveloper のみです。次に、Oracle JDeveloper の特徴を示します。

- ブレーク・ポイント、監視式、インスペクタを含むグラフィカルなリモート・デバッガ
- マルチ・ドキュメント・インタフェース (MDI)
- *Codecoach* 機能を利用したコードの最適化
- 専用コードまたはマーカーを使用せずに、完全に純粋な Java アプリケーション、アプレット、サーブレット、JavaBeans などの生成が可能
- Oracle Database ブラウザ

注意： Oracle JDeveloper は、Oracle RDBMS にはパッケージされていないアプリケーションです。

BI Beans の概要

BI Beans では、OLAP 意思決定支援アプリケーションの基本的なビルディング・ブロックである再利用可能なコンポーネントが提供されます。BI Beans の大規模な機能単位はすでに開発され、その堅牢性および使いやすさがテスト済であるため、BI Beans を使用すると、開発者は新しいアプリケーションを迅速に開発し、配置できます。また、BI Beans では OLAP アプリケーションに共通のルック・アンド・フィールが提供されるため、エンド・ユーザーの学習曲線が大幅に短縮されます。

BI Beans には次のものが含まれます。

- **プレゼンテーション Beans** では、傾向と変化を簡単に検出できるように、様々な形式でデータを表示します。現在、プレゼンテーション Beans ではグラフ、表およびクロス集計が使用可能です。
- **データ Beans** はデータを取得し、操作します。データ Beans は OLAP API を使用してデータソースに接続し、問合せを定義して結果データ・セットを操作した後、結果をプレゼンテーション Beans に戻して表示します。データ Beans には、Query Builder および Calculation Builder が含まれます。
- **永続サービス** は、BI Beans カタログ内のオブジェクトの格納および取得をサポートするパッケージのセットです。これによって作業を保存できるだけでなく、このカタログに対するアクセス権の所有者と作業を共有することもできます。

Java クライアントまたは Thin クライアントとして BI Beans を実装することもできます。Java クライアントは、分析に特化した（長期間にわたり、システムに対して多数の対話を実行する）ユーザーに適しています。たとえば、レポートを作成するユーザーは Java クライアントを使用すると効果的です。Thin クライアントは、低帯域幅接続を使用し、基本分析を行うリモート・ユーザーに適しています。Thin クライアントは、このようなユーザー向けに、ポータルまたはその他の Web サイトに埋め込むことができます。

メタデータ

OLAP API および BI Beans は、OLAP カタログを使用して、Oracle データ・ウェアハウスで定義された多次元オブジェクト（メジャーやディメンションなど）について必要な情報を提供します。BI Beans は追加のメタデータを生成し、追加の機能をサポートします。この拡張バージョンを BI Beans カタログといいます。

ナビゲーション

プレゼンテーション Beans では、ドリル、ピボット、ページングなどのナビゲーション方法をサポートしています。

- ドリルは、州を構成する市など、上位レベルの集計を構成する下位レベルの値を表示します。
- ピボットは、シリーズというラベルを持つディメンション・メンバーがグループというラベルになるように、また、クロス集計で列というラベルを持つディメンション・メンバーが行というラベルになるように、データ・キューブを回転させます。たとえば、製品のラベルが行で地域のラベルが列の場合、製品が列、地域が行になるようにデータ・キューブを回転させることができます。
- ページングは、各メンバーを行または列にネストするのではなく、別々のグラフ、クロス集計または表に表示することによって、追加ディメンションを処理します。たとえば、同じグラフにすべての期間を表示するのではなく、別々のグラフに各期間を表示する場合などがあります。

書式設定

プレゼンテーション Beans を使用すると、特定の表示方法を変更できます。また、データの値自体が書式に影響する場合があります。

- 数値書式設定。数値の表示は、スケール、小数点以下の桁数、先行ゼロ、通貨記号、負数の表記法などを変更することによって変更できます。通貨記号およびスケール要素は、セルではなく、列または行ヘッダーに表示できます。
- スポットライト書式設定。異常値や問題のある結果を他のデータ値と区別できるように、セルの背景色、罫線、フォントなどの書式をデータ・ドリブンにできます。
- ランキング。ランキング・レポートでは、メジャー値に基づく各ディメンション値の数値ランクが表示されます。

グラフ

Graph Bean は、棒、面、線、円、ドーナツ、散布図、バブル、ピラミッド、株価など、2次元および3次元の様々なビジネス・グラフでデータを表示します。2D グラフの多くは、結合、積上げ、2軸、パーセント、水平、垂直または3D 効果グラフとして表示できます。

棒、線および面グラフは、データ・キューブの各行をいずれか 1 種類のグラフとして指定できるように組み合わせることができます。また、行ごとにマーカーの形と種類、データのフォント、色、幅および塗りつぶしの色を指定することもできます。

グラフのイメージは、システムのクリップボードにコピーでき、GIF などのイメージ書式でエクスポートできます。

グラフで選択した領域は、ズームインまたはズームアウトが可能です。また、軸間でスクロールすることもできます。

クロス集計

Crosstab Bean は、スプレッドシートに似た 2 次元のグリッドでデータを表示します。複数のディメンションを行または列に沿ってネストでき、追加のディメンションを別のページとして表示できます。カスタマイズが可能な項目は、フォント、サイズ、色、下線、各セルの背景色、罫線の書式およびテキストの位置合せ行揃えです。

マウスまたはキーボードを使用してデータ間をナビゲートできます。合計を表示する行および列を挿入したり、what-if 分析用にセルを編集することが可能です。

表

Table Bean は、リレーショナル表またはリレーショナル・ビューと同様のレコード形式でデータを表示します。クロス集計とは異なり、表表示では、メジャーはメジャー・ディメンションのメンバーとしてでなく、個別に処理されます。つまり、各メジャーを個々に操作することが可能です。

データ Beans

データ Beans は、OLAP API を使用して、アプリケーションに必要な基本的なサービスを提供します。データ Beans を使用すると、クライアントでのデータベースの識別、そのデータベースにアクセスするための証明書の表示、および接続の確立が可能になります。これによって、アプリケーションはメタデータにアクセスし、使用可能なデータを識別できるようになります。ユーザーは、参照したいメジャー、および必要なデータの特定の部分を選択できます。その後、そのデータの変更および操作が可能になります。

ウィザード

BI Beans では、アプリケーション開発者が初期環境を作成する場合、およびエンド・ユーザーがそれぞれのニーズに応じてアプリケーションをカスタマイズする場合に使用できるウィザードが提供されます。このウィザードに従って操作すると、アプリケーションに必要なすべての情報を指定できます。次に、ウィザードで実行できるタスクを示します。

- 問合せの作成。ファクト表およびマテリアライズド・ビューには、ユーザーが表示したいデータよりはるかに多くのデータが含まれている場合があります。また、大量のデータをフェッチすると、パフォーマンスが大幅に低下する可能性があります。メジャーを

選択する他に、リストからディメンション・メンバーを選択したり、条件を使用して、問合せでフェッチするデータの量を制限することができます。選択結果は保存でき、リストから名前を選択するだけで再利用できます。

BI Beans では、ランキング、ラグ、リード、ウィンドウなど、データベースの新しい SQL 分析関数をすべて利用します。SQL を知らないエンド・ユーザーでも、高度な分析が必要な強力な問合せを作成できます。

- カスタム・メジャーの生成。データベースに格納されたデータから値を計算する「カスタム」メジャーを新しく定義できます。たとえば、1 年前からの売上げの変化率を表示するカスタム・メジャーを作成する場合などがあります。カスタム・メジャーのデータは、Sales メジャーのデータに対してラグ・メソッドを使用して計算されます。DBA がすべてのユーザーに必要なすべての計算を予想して作成することはできないため、BI Beans を使用して、ユーザーが独自に作成できます。

OLAP API の理解

通常、OLAP アプリケーションには、オブジェクト指向のユーザー・インタフェースがあります。ユーザーは、ユーザー・インタフェースを使用して、データをグループ化して表示するオブジェクトを操作します。つまり、オブジェクト指向のユーザー・インタフェースとオブジェクト指向の API (Oracle OLAP API) の間には当然の関連性があります。OLAP API は、アプリケーションに必要なエンド・ユーザーの動作と一致するオブジェクトを指定することによってこの関係を利用します。

Java などのオブジェクト指向言語は、オブジェクトにメソッドを適用することでデータを操作します。この方法によって、オブジェクトは現在の状態を維持でき、現在の状態への増分変更がサポートされます。この方法では、ドリルや回転など、共通の OLAP アクションがサポートされます。

たとえば、OLAP アプリケーションのユーザーの主な動作は問合せの改良です。ユーザーは、疑問を解決するための問合せを作成します。ほとんどの場合、さらに詳細なデータが表示されるようにドリルしたり、レポートを回転させてデータの相関を明らかにするなどして、解決方法を探るように求めるプロンプトが問合せの初期結果として表示されます。OLAP API では、いずれかの問合せの結果を次の問合せの入力として使用できます。

OLAP API による多次元データへのアクセス方法

OLAP API は、OLAP カタログ (OLAP メタデータを含むリレーショナル表) を使用してデータにアクセスします。アプリケーションでは、データがリレーショナル表またはアナリティック・ワークスペースのいずれにあるか、およびデータへのアクセス方法を認識する必要があります。

Oracle OLAP では、すべての問合せを OLAP API から SQL に変換します。OLAP API から問合せが発行されると、Oracle OLAP の SQL Generator がリレーショナル表またはリレーショナル・ビューに SELECT 文を発行します。これによって、アプリケーション開発者には次のメリットがあります。

- 多次元問合せを解決するために必要な複雑な SQL を書くという困難な作業、およびその SQL を最適化するというさらに困難な作業は Oracle OLAP が行います。アプリケーション開発者は OLAP 専用の OLAP API で、より生産的に SQL を書くことができます。
- SQL および OLAP DML の更新版が、新しいバージョンの OLAP API に組み込まれます。アプリケーションでは、履歴なしで新しい分析機能およびパフォーマンス機能を使用できます。

また、OLAP API では、Java アプリケーションを使用してアナリティック・ワークスペースのデータを直接操作する方法も提供しています。この場合、メタデータは不要で、OLAP API のデータ操作クラスを使用する必要もありません。Java アプリケーションは、OLAP API の `SPLExecutor` クラスを使用して OLAP DML コマンドを直接 Oracle OLAP に送信し、アナリティック・ワークスペースで実行します。

どのアクセス方法を使用した場合も、アプリケーションでは接続を確立し、アナリティック・ワークスペースを開き、データにアクセス（MDM メタデータまたは `SPLExecutor` のいずれかを使用）した後、アナリティック・ワークスペースを閉じ、接続を終了します。

参照：

- 『Oracle OLAP 開発者ガイド - Oracle OLAP API』を参照してください。
- OLAP API Javadoc を参照してください。

インテリジェント・キャッシング

基本的に分析問合せは繰り返されます。分析者は問合せを作成し、結果を確認した後、その問合せに基づいて他の問合せを作成します。ビジネス分析では、後の問合せに対する回答で同じデータが必要になる可能性が非常に高いため、OLAP API ではメタデータを再度フェッチしなくてもセッション期間中に使用できるように、そのメタデータをキャッシュします。また、OLAP API は問合せの結果セットを幾何学的に定義します。OLAP API では、多次元カーソルを使用して、結果セットの様々な領域にランダムにアクセスできます。これによって、アプリケーションは結果セットのすべてのデータではなく、現在必要なデータのみを取得できます。たとえば、あるページ上のすべてのデータをフェッチせずに、そのページの末尾にスクロールする場合などです。

データ・ウェアハウスからデータを取得するために、OLAP API では SQL 文を生成します。データ・フェッチでは、連結ロールアップ、スクロール可能カーソルおよびクエリー・リライトを含む、Oracle Database の多くの最新機能を使用します。

計算機能

OLAP API では、SQL コマンドを生成して、リレーショナル表に格納されたデータを選択および操作します。これらの SQL コマンドには、`RANK`、`PERCENTILE`、`TOPN`、`BOTTOMN`、`LAG`、`LEAD`、`SUM`、`AVG`、`MIN`、`MAX`、`COUNT`、`STDDEV` などの SQL 分析関数が含まれます。

OLAP API では、その他の OLAP ソリューションでは効率的に処理できる機能を超える、次のような拡張計算機能が提供されます。

- 複数の属性による合計の算出
- NA および 0（ゼロ）の行、列およびページの非表示
- 結合ディメンション
- ディメンションとしてのメジャー
- 次の出荷受注比率などの行間計算

`Balance(Account "BOOKED", Period "PRIOR") / Balance(Account "BILLED", Period "LAST")`

- 非対称問合せ

OLAP エンジンでは、次のような追加の計算を実行します。

- モデリング
- 予測
- what-if シナリオ

このような分析は、アナリティック・ワークスペースのデータに対して実行できます。

第 II 部

アナリティック・ワークスペースの作成および使用の基礎

第 II 部には、アナリティック・ワークスペースの作成に関する情報が含まれます。ここでは、次の項目について説明します。

- [第 5 章「OLAP カタログ・メタデータの作成」](#)
- [第 6 章「アナリティック・ワークスペースの作成」](#)
- [第 7 章「アナリティック・ワークスペースへの SQL アクセス」](#)

OLAP カタログ・メタデータの作成

この章では、OLAP カタログおよび OLAP メタデータでの操作方法を説明します。この章では、次の項目について説明します。

- OLAP カタログの概要
- OLAP カタログ・メタデータの作成用ツールの選択
- Oracle Enterprise Manager を使用したメタデータの作成
- 事歴 : GLOBAL スター・スキーマのメタデータ作成
- PL/SQL を使用したメタデータの作成

OLAP カタログの概要

OLAP カタログは論理多次元オブジェクトを定義し、それらを物理データソースにマップします。論理オブジェクトとは、2-1 ページの「**論理多次元データ・モデル**」で説明しているキューブ、メジャー、ディメンションなどです。物理データソースはリレーショナル表またはリレーショナル・ビューの列です。OLAP カタログ・メタデータによって様々なウェアハウス構成を表示できます。

OLAP カタログは、アナリティック・ワークスペースに対して次の別個の機能を提供します。

- スター・スキーマまたはスノーフレーク・スキーマのリレーショナル表を記述し、アナリティック・ワークスペースにデータを変換できるようにする。このメタデータは、アナリティック・ワークスペースを構築またはリフレッシュする際にのみ使用されます。
- アナリティック・ワークスペースのリレーショナル・ビューを記述し、BI Beans または OLAP API でデータを問い合わせできるようにする。このメタデータは、アプリケーションでワークスペース・データへアクセスできるように、実行時にのみ使用されます。

したがって、アナリティック・ワークスペースを開発する場合、ソース・スキーマ用とアナリティック・ワークスペース用に 1 つずつ、2 つの OLAP カタログ・メタデータのセットを

作成します。アナリティック・ワークスペースを **Oracle Discoverer** で使用する場合、ソース・スキーマ用の OLAP カタログ・メタデータのみを定義します。

OLAP カタログは、BI Beans または OLAP API でデータを問い合わせできるように、スター・スキーマのリレーショナル表の記述にも使用されます。このタイプのシナリオでは、アナリティック・ワークスペースは使用されず、集計データはマテリアライズド・ビューに格納されます（[第 13 章](#)を参照）。

BI Beans および OLAP API では、OLAP カタログに格納されているメタデータを問い合わせます。データは、リレーショナル表またはアナリティック・ワークスペースのいずれに格納されていても、OLAP カタログで識別されないかぎり、これらのテクノロジーでアプリケーションからアクセスすることはできません。OLAP カタログは、どのようなアプリケーションでも使用できます。

OLAP カタログ・コンポーネント

OLAP カタログには、次のコンポーネントが含まれます。

- **メタデータ・モデル表：** OLAP メタデータ・モデルをインスタンス化する、データベース内のリレーショナル表の集合。これらの表は、すべての OLAP メタデータ・オブジェクト（ディメンション、メジャー、キューブ、メジャー・フォルダなど）を定義します。メタデータでは、定義は実際のデータソースへの参照です。
- **書込み API：** OLAP メタデータを作成および編集するための PL/SQL パッケージの集合。これらのパッケージには、モデル表の行を挿入、更新および削除するためのプロシージャが含まれています。
- **読み込み API：** モデル表に登録されたメタデータに関する情報を提供する、データベース内のリレーショナル・ビューの集合。

現在使用される OLAP カタログには、CWM1（CWM-Lite とも呼ばれる）および CWM2 の 2 つのバージョンがあります。各バージョンには独自のメタデータ・モデル表、書込み API および読み込み API があります。ただし、アプリケーションでは、メタデータの生成に使用される書込み API にかかわらず、すべての OLAP カタログ・メタデータを含む UNION ビューの集合を問い合わせることができます。

CWM1 について

CWM1 は、Oracle Enterprise Manager の OLAP 管理ツールおよび Oracle Warehouse Builder リリース 9.2 で使用できます。5-3 ページの「[OLAP カタログ・メタデータの作成用ツールの選択](#)」に示す要件を満たすスキーマを定義するためだけに CWM1 を使用できます。その後で、OLAP カタログを使用してアナリティック・ワークスペースを作成したり、OLAP API または BI Beans でリレーショナル・スキーマに直接アクセスしたりできます。

Enterprise Manager の OLAP 管理ツールまたは Analytic Workspace Manager の OLAP カタログ・ビューで、CWM1 メタデータを表示できます。

CWM2 について

CWM2 は、Analytic Workspace Manager の OLAP API イネーブラを介して、および PL/SQL パッケージのセットとして使用できます。CWM2 を使用して、CWM1 の要件を満たさないスター・スキーマまたはスノーflake・スキーマを定義できます。アナリティック・ワークスペースのメタデータの定義に使用できるのは CWM2 のみで、この目的で CWM1 を使用することはできません。

Analytic Workspace Manager の OLAP カタログ・ビューで CWM2 メタデータを表示できます。

OLAP メタデータを作成するための手順

OLAP メタデータの作成にプログラムを使用するか GUI を使用するにかかわらず、次の同じ基本手順を実行します。

次の手順で OLAP メタデータを作成します。

1. 論理ディメンションを作成します。各ディメンションに関連付けられているレベル、属性および階層を指定します (5-10 ページの「[手順: OLAP カタログでの論理ディメンションの定義](#)」を参照)。
2. 論理キューブを作成し、そのエッジ (ディメンション) を指定します (5-11 ページの「[手順: OLAP カタログでの論理キューブの定義](#)」を参照)。
3. ファクト・データを表示する論理メジャーを作成します。各メジャーをキューブに関連付けます (「[手順: OLAP カタログでの論理キューブの定義](#)」を参照)。
4. 論理エンティティをソース・データにマップします (「[手順: OLAP カタログでの論理キューブの定義](#)」を参照)。

OLAP カタログ・メタデータの作成用ツールの選択

OLAP カタログ・メタデータの作成用ツールは、リレーショナル・スキーマとアナリティック・ワークスペースのどちらのメタデータを作成するかによって異なります。いくつかのツールには、固有の前提条件があります。

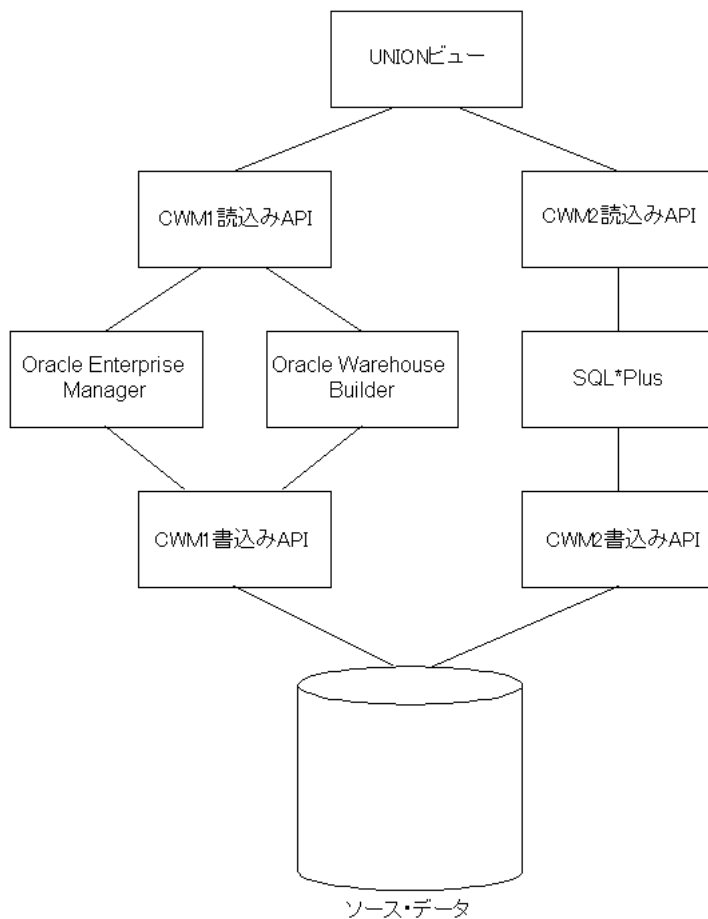
ソース・データのメタデータの作成

リレーショナル・スキーマの OLAP カタログ・メタデータを作成するには、次の 3 つのツールから選択できます。

- Oracle Enterprise Manager リリース 9.2
- Oracle Warehouse Builder リリース 9.2
- CWM2 PL/SQL API

CWM1 PL/SQL API は、GUI でのみ使用できます。図 5-1 に、これらのツールとソース・データ間の関係を示します。

図 5-1 ソース・データのメタデータの作成用ツール



この章では、Oracle Enterprise Manager の OLAP 管理ツールについて説明しています。

参照： CWM2 API の完全な構文および詳細については、『Oracle OLAP リファレンス』を参照してください。

Warehouse Builder を使用したアナリティック・ワークスペースの作成手順については、『Oracle Warehouse Builder ユーザーズ・ガイド』を参照してください。

基本的なスター・スキーマまたはスノーフレイク・スキーマ

OLAP 管理ツールで使用される CWM1 書込み API は、各論理 OLAP ディメンションのデータベース・ディメンション・オブジェクトを作成します。スター・スキーマまたはスノーフレイク・スキーマのディメンション表および関連するファクト表には、データベースのディメンション・オブジェクトによって次の制限事項が課せられます。

- すべての階層はレベルベースである必要があります。スキーマでは親子ディメンション表を使用できません。
- ディメンションに対して定義された複数の階層には、同じベース・レベルが必要です。
- レベル列に NULL を含めることはできません。
- ファクト・データは未解決である必要があります。つまり、ファクト・データは階層の最低レベルに格納され、キューブのすべてのデータは単一のファクト表に格納される必要があります。

ソース・データがスター・スキーマまたはスノーフレイク・スキーマで、これらの追加要件を満たしている場合、使用している環境に応じて Oracle Enterprise Manager または CWM2 API を使用できます。Oracle Enterprise Manager の OLAP 管理ツールでは、GUI が提供されます。CWM2 API を使用すると、変更や他のデータベースへの移植が簡単にできる SQL プログラムを作成できます。

ソース・データがスター・スキーマまたはスノーフレイク・スキーマで、これらの要件を満たしていない場合、CWM2 API を使用します。

複雑な階層を持つディメンション表

ソース・データがスター・スキーマまたはスノーフレイク・スキーマであるが、次のバリエーションのいずれかがディメンション表に含まれる場合は、CWM2 API を使用します。

- NULL を含むレベル列（スキップレベルの階層など）
- 異なるベース・レベルを持つ複数階層（**不規則階層**とも呼ぶ）
- 異なるレベルにマップされる値を持つ複数階層
- 親子ディメンション

スキーマに親子ディメンション表が含まれる場合、それらをレベル・ベースのディメンション表に変換する必要があります。CWM2 書込み API には、この変換用のパッケージが含まれます。

その他のスキーマ構成

ソース・データがスター・スキーマまたはスノーフレイク・スキーマでないか、他の変換を必要とする場合、Oracle Warehouse Builder の OLAP Bridge を使用します。このツールを使用すると、スター・スキーマの生成、OLAP カタログ・メタデータにおける論理多次元データ・モデルの定義、アナリティック・ワークスペースの作成および移入、OLAP API または BI Beans によるワークスペースの使用の有効化を実行できます。

アナリティック・ワークスペースのメタデータの作成

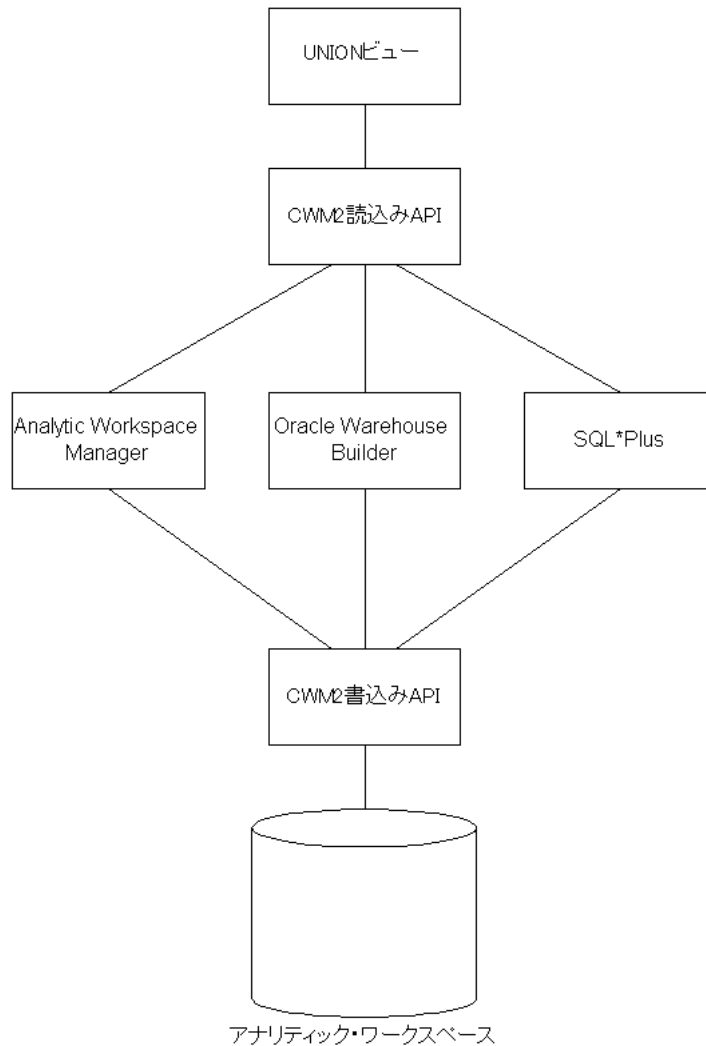
アナリティック・ワークスペースに格納されているデータの OLAP カタログ・メタデータを作成する場合、ワークスペースの多次元オブジェクトのリレーショナル・ビューに対して定義します。これらのビューはスター・スキーマを模倣しますが、ディメンションを表示する方法が異なります。ディメンション階層の各レベルに個別の列を表示するのではなく、これらのビューは、単一列のすべてのレベルのディメンション・メンバーをすべて表示します。このような理由から、このタイプのビューを**埋込み合計**ビューと呼びます。ビューは、アナリティック・ワークスペースに格納される同じ形式でディメンションを表示します。埋込み合計ビューの CWM2 メタデータを作成することはできますが、CWM1 メタデータは作成できません。

アナリティック・ワークスペースの OLAP カタログ・メタデータを作成するには、次の3つのツールから選択できます。

- Analytic Workspace Manager
- Oracle Warehouse Builder
- CWM2 API

 図 5-2 に、これらのツールとアナリティック・ワークスペース間の関係を示します。

図 5-2 アナリティック・ワークスペースのメタデータの作成用ツール



スタンダード・フォームのアナリティック・ワークスペースの場合、Analytic Workspace Manager の OLAP API および BI Beans イネーブラを使用して、リレーショナル・ビューおよび CWM2 メタデータを 1 回の手順で生成します。

アナリティック・ワークスペースの生成に Oracle Warehouse Builder を使用した場合、OLAP API および BI Beans でアクセス可能なビューおよび CWM2 メタデータも作成されています。

アナリティック・ワークスペースにデータベース・スタンダード・フォームに準拠しないオブジェクトが含まれているか、リレーショナル・ビューを手動で生成する場合は、独自の CWM2 スクリプトを作成します。Analytic Workspace Manager で生成したスクリプトを修正するところから始めることもできます。

第 6 章「アナリティック・ワークスペースの作成」に、Analytic Workspace Manager の OLAP API および BI Beans イネーブラの使用方法を示します。

参照： CWM2 API の完全な構文および詳細については、『Oracle OLAP リファレンス』を参照してください。

Warehouse Builder を使用したアナリティック・ワークスペースの作成手順については、『Oracle Warehouse Builder ユーザーズ・ガイド』を参照してください。

Oracle Enterprise Manager を使用したメタデータの作成

5-5 ページの「[基本的なスター・スキーマまたはスノーフレーク・スキーマ](#)」に示した要件をデータ・ウェアハウスが満たす場合は、Oracle Enterprise Manager で OLAP 管理ツールを使用して OLAP メタデータを作成できます。

主にウィザード形式の手順を実行するか、またはプロパティ・シートを記入して、メタデータを作成する SQL 文を生成します。必要に応じて、実行前に SQL 文を表示できます。

注意： CWM1 コールのスクリプトを保存するには、「Configuration」メニューから「SQL Logging」を選択します。ログ・ファイルをスクリプトとして実行する前に、編集してセミコロン (;) およびスラッシュ (/) などのコマンド終了記号を追加する必要があります。

手順 : OLAP 管理へのアクセス

次の手順を実行して Oracle Enterprise Manager を起動し、OLAP 管理にアクセスします。

1. Oracle Enterprise Manager コンソールを開きます。

メイン・ページが表示されます。

2. プラス記号をクリックして「Databases」を開きます。

接続を定義した Oracle Database のサービス名のリストが表示されます。

管理するデータベースが表示されない場合は、「Navigator」メニューから「Add Database to Tree」を選択します。ホスト名、ポート番号および SID を指定します。

3. 管理するデータベースを開きます。

「Database Connect Information」ダイアログ・ボックスが表示されます。

4. そのデータベースに対するユーザー名（適切な資格証明を得ているもの）およびパスワードを入力します。

ヒント： 以降のセッションでこの手順を省略する場合は、「**Save as preferred credentials**」ボックスを選択します。ユーザー名および暗号化されたパスワードはローカル・ファイルに保存されます。セキュリティ上、格納された資格証明で Oracle Enterprise Manager を実行できるのは、この手順でユーザー名およびパスワードを入力したユーザーのみにしてください。後でこの情報を変更する場合は、「**Configuration**」メニューから「**Edit Local Preferred Credentials**」を選択してください。

データベース・フォルダが開き、データベースを管理できる様々なツールが表示されます。

5. 「**Warehouse**」を開きます。
6. 「**OLAP**」を開きます。

作成できるオブジェクトのタイプが表示されます。このフォルダが OLAP 管理用です。

ディメンション表へのメタデータの定義

OLAP メタデータを作成する場合は、まずディメンション表にメタデータ・オブジェクトを定義する必要があります。これらのメタデータ・オブジェクトは、データベースのディメンション・オブジェクトに基づく論理ディメンションです。ディメンション作成ウィザードを使用するか、または「**Create Dimension**」ダイアログ・ボックスに直接情報を指定できます。

ディメンションに指定する情報

ディメンションを定義するには、それによってディメンション化されたメジャーのラベル付けおよび集計に必要な次の情報をすべて指定します。

- ディメンションの名前
- ディメンションのデータを含む表
- 各レベルの名前、および各レベルのデータを含む列
- 各階層のレベルの数および順序
- 別々の表に格納されているレベルの結合キー
- レベルの属性を含む列
- ディメンションとその階層、レベルおよび属性の表示名および定義

時間ディメンション

ビジネス分析は履歴データ上で実行されるため、完全に定義された期間が必須です。時間ディメンション表には期間の最終日および期間の列が必要です。この情報によって、前の期間との比較など、時系列の分析がサポートされます。スキーマにこれらの列がない場合も時間を通常のディメンションとして定義できますが、時間ベースの分析はサポートされません。

Time ディメンションの通常のレベルおよび階層はディメンション・ウィザードで提案されますが、これらを使用する必要はありません。

手順 : OLAP カタログでの論理ディメンションの定義

次の手順を実行して、ディメンションおよび関連付けられたレベル、階層および属性を作成します。

1. Oracle Enterprise Manager を起動し、OLAP 管理にアクセスします (5-8 ページの「[手順 : OLAP 管理へのアクセス](#)」を参照)。
2. 「Dimensions」を右クリックし、次のいずれかを選択します。
 - ディメンション・ウィザードを実行する場合は「Create Using Wizard」
または
 - 新しいディメンションのプロパティ・シートを編集する場合は「Create」
3. 追加情報が必要な場合は、「Help」を選択します。

ファクト表へのメタデータの定義

ディメンション表にメタデータ・オブジェクトを定義した後、ファクト表にメタデータ・オブジェクトを作成できます。これらのメタデータ・オブジェクトはメジャーおよびキューブです。キューブは、同様にディメンション化されたメジャーの集合です。キューブおよびメジャー全体が OLAP メタデータで定義されます。対応するデータベース・オブジェクトはありません。

ヒント： 予測などのカスタム・メジャーをアナリティック・ワークスペースの永続的な部分として計算および格納する場合、空の列をファクト表に追加し、これらの列から論理メジャーを定義できます。次に、アナリティック・ワークスペースの作成プロセスによって、そのカスタム・メジャーに関連付けられるすべてのオブジェクトが作成および登録されるので、あとはそれらを移入するのみです。カスタム・メジャーの作成方法の詳細は、[第 11 章](#)を参照してください。

キューブに指定する情報

キューブを定義する場合は、次のような情報を指定します。

- キューブおよびそれに関連付けられたファクト表の名前。キューブのすべてのメジャーは、単一のファクト表からのものである必要があります。
- キューブで使用するディメンションおよびディメンション階層のレベルの名前。
- メジャーおよび各メジャーの値が格納されるファクト表の列の名前。
- 各メジャーの各ディメンションに対するデフォルトの集計演算子（sum や average など）。
- 計算の依存関係。

手順：OLAP カタログでの論理キューブの定義

次の手順でキューブを作成します。

1. Oracle Enterprise Manager を起動し、OLAP 管理にアクセスします（5-8 ページの「[手順：OLAP 管理へのアクセス](#)」を参照）。
2. 「Cubes」を右クリックし、次のいずれかを選択します。
 - キューブ・ウィザードを実行する場合は「**Create Using Wizard**」または
 - 新しいキューブのプロパティ・シートを編集する場合は「**Create**」
サマリー・アドバイザーは実行しないでください。
3. 追加情報が必要な場合は、「**Help**」を選択します。

キューブのデータの表示

Cube Viewer を使用すると、作成したキューブが、エンド・ユーザーに表示されるのと同じ状態（BI Beans クロス集計にデータが表示される）で表示されます（4-5 ページの「[クロス集計](#)」を参照）。また、Query Builder を使用すると、表示するデータを選択できます。

注意： Cube Viewer で表示できるのは、Oracle Enterprise Manager で作成したキューブのみです。

次の手順でキューブを表示します。

1. Oracle Enterprise Manager を起動し、OLAP 管理にアクセスします（5-8 ページの「[手順：OLAP 管理へのアクセス](#)」を参照）。
2. キューブのリストを確認できるように OLAP ツリーを開きます。

3. 確認するキューブを右クリックし、「**Cube Viewer**」を選択します。
4. データの選択を変更するには、「File」メニューから「**Query Builder**」を選択します。
5. 追加情報が必要な場合は、Cube Viewer の「Help」メニューからオプションを選択します。

事歴 : GLOBAL スター・スキーマのメタデータ作成

Global スター・スキーマは CWM1 のすべての要件を満たすので、Oracle Enterprise Manager の OLAP 管理ツールを使用できます。

Global スキーマをインストール済の場合、OLAP カタログ・メタデータはすでに定義されています。ただし、この例では、異なるスキーマのメタデータを作成します。論理オブジェクトとソース列との間のマッピングについてはすべて、[第 3 章](#)で説明しています。次の手順では、ディメンションおよびキューブを 1 つずつ定義する方法について説明します。

Global スキーマの論理 Time ディメンションの定義

TIMES_DIM 表は、3 つのレベル (Month、Quarter および Year) を持つ単一の Calendar 階層をサポートしています (3-9 ページの「[ディメンション表 : TIME_DIM](#)」を参照)。次に示す手順では、ディメンション作成ウィザードを使用して論理 Time ディメンションを定義します。

1. 「Choose Dimension Type」ページで、「**Create a time dimension object**」および「**Based on an existing lookup table**」を選択します。
2. 「Name and Schema」ページで、名前として「TIME」を入力し、「GLOBAL」スキーマを選択します。
3. 「Define Levels」ページで、次の手順を実行します。
 - a. 右側の「**Properties**」セクションで、「TIME_DIM」表を選択します。
 - b. 左側の「**Levels**」セクションで、「Year」を選択します。次に、「**Available Columns**」から「YEAR_ID」を選択し、矢印キーをクリックして「**Selected Columns**」まで移動させます。
 - c. 「Quarter」を選択し、次の同じ手順に従って、「QUARTER_ID」列にマップします。
 - d. 「Month」を「MONTH_ID」列にマップします。
 - e. 「Day」を選択し、「**Delete**」をクリックします。
4. 「Define Attributes」ページで、次の手順を実行します。
 - a. 左側の「**Attributes**」セクションから「Long_Description」を選択します。

- b. 各レベルに対して、「Attribute Source Column」下の適切な列を選択します。セルをクリックして選択のリストを表示します。

各属性に対してこれらの手順を繰り返し、値をマップします (3-16 ページの「[Global Time ディメンションのマッピング](#)」を参照)。

5. 「Define Hierarchies」ページで、「Calendar」階層を選択します。次に使用可能なすべてのレベルを選択し、次のような階層を形成するようにします。

```
Year
|_Quarter
   |_Month
```

6. 「Fiscal」階層を選択し、「Delete」をクリックします。
7. 「OLAP Options」ページで、追加したい説明を入力します。

ウィザードで Time ディメンションの作成が完了したら、「Dimensions」下の「GLOBAL」ツリーを開いて表示できます。

Global スキーマの論理 Units キューブの定義

UNITS_HISTORY_FACT 表は、4 つのディメンション表からの 4 つのサロゲート・キーから構成される複数列の主キー、および 1 つのメジャー (UNITS) を持ちます。次に示す手順では、キューブ作成ウィザードを使用して論理 Units キューブを定義します。

1. 「Provide General Information」ページで、名前として「UNITS_CUBE」を入力します。
Global スキーマをインストール済の場合、このキューブはすでに定義されています。ただし、この手順では、異なる名前または異なるスキーマのキューブを作成します。
2. 「Choose a Fact Table」ページで、「Global」ノードを開いて「UNITS_HISTORY_FACT」を選択し、矢印をクリックして「Selected Table」ペインに移動させます。
3. 「Add Dimensions」ページで「Global」ノードを開いて、4 つのディメンション (CHANNEL、CUSTOMER、PRODUCT および TIME) をすべて選択し、矢印をクリックしてこれらのディメンションを「Selected Dimensions」ペインに移動させます。
4. 「Specify Dimension Properties」ページで、左側の「Dimension」ペインの各ディメンションに対して次の手順を実行します。
適切な外部キー列を選択します。ファクト表の外部キー列のセルをクリックすると、ファクト表の列のリストが表示されます。この場合、ディメンション表のキー列とファクト表の外部キーは同じ名前です。
5. 「Specify Measures」ページのデフォルト設定を受け入れます。
6. サマリー・アドバイザは実行しないでください。「Summary」ページで、サマリー・アドバイザ・ウィザードを起動するためのチェック・ボックスのチェックを外します。チェックを外さなかった場合も、サマリー・アドバイザの「Welcome」ページから取り消すことができます。

7. SQL*Plus Worksheet を開き、次のコマンドを発行します。

```
EXECUTE CWM2_OLAP_METADATA_REFRESH.MR_REFRESH
```

PL/SQL を使用したメタデータの作成

PL/SQL パッケージ CWM2 には、様々なスキーマ設計の OLAP メタデータを作成できるストア・プロシージャが含まれています (5-3 ページの「[OLAP カタログ・メタデータの作成用ツールの選択](#)」を参照)。

これらのパッケージを使用する前に、必要な準備作業が完了していることを確認します。

参照：

- PL/SQL パッケージ CWM2 を使用する完全な例については、B-4 ページの「[OLAP カタログ・メタデータ用の SQL スクリプト](#)」を参照してください。
- CWM2 パッケージの包括的な構文については、『Oracle OLAP リファレンス』を参照してください。

OLAP ディメンションを作成するための CWM2 パッケージ

次のパッケージには、ディメンション表にメタデータを作成するプロシージャが含まれています。

- CWM2_OLAP_DIMENSION には、ディメンションを作成するプロシージャが含まれています。
- CWM2_OLAP_HIERARCHY には、ディメンションの階層を作成するプロシージャが含まれています。
- CWM2_OLAP_LEVEL には、ディメンションのレベルを作成し、レベルを階層に関連付けるプロシージャが含まれています。
- CWM2_OLAP_LEVEL_ATTRIBUTE には、レベル属性を作成し、それらをレベルに関連付けるプロシージャが含まれています。
- CWM2_OLAP_DIMENSION_ATTRIBUTE には、ディメンション属性を作成し、それらをディメンションに関連付けるプロシージャが含まれています。

キューブを作成するための CWM2 パッケージ

次のパッケージには、ファクト表にメタデータを作成するプロシージャが含まれています。

- CWM2_OLAP_CUBE には、キューブの多次元構造を作成するプロシージャが含まれています。

- CWM2_OLAP_MEASURE には、メジャーを作成し、それらをキューブに関連付けるプロシージャが含まれています。

メタデータをマッピングするための CWM2 パッケージ

CWM2_OLAP_TABLE_MAP パッケージには、論理メタデータ・エンティティを物理データソースにマップするプロシージャが含まれています。データはリレーショナル表に格納されるか、またはリレーショナル・ビューに表示されます。ディメンション表およびファクト表がビューとして定義されている場合、実際のデータはアナリティック・ワークスペースにあります。

レベルベース・ディメンション表を作成するための CWM2 パッケージ

CWM2_OLAP_PC_TRANSFORM パッケージには、親子ディメンション表をレベルベース・ディメンション表に変換するプロシージャが含まれています。OLAP API でディメンションにアクセスする場合、この変換は必須です。

分類および検証のための CWM2 パッケージ

次のパッケージには、メジャー・フォルダを作成し、OLAP メタデータを検証するプロシージャが含まれています。

- CWM2_OLAP_CATALOG では、メジャー・フォルダを作成およびメンテナンスするプロシージャが提供されます。
- CWM2_OLAP_VALIDATE パッケージでは、OLAP カタログ・メタデータを検証するプロシージャが提供されます。
- CWM2_OLAP_METADATA_REFRESH では、OLAP API による問合せをサポートしているメタデータ表をリフレッシュするプロシージャが提供されます。

アナリティック・ワークスペースの作成

この章では、最適に実行されるスタンダード・フォームのアナリティック・ワークスペースの作成方法について説明します。データをアナリティック・ワークスペースにどのように格納し、どれだけ迅速にそのデータを取得できるのか、ということに影響する設計における様々な決定についても調査します。これらの決定については、Analytic Workspace Managerのウィザードを使用するコンテキストで説明します。

この章では、次の項目について説明します。

- [アナリティック・ワークスペースの作成方法](#)
- [Analytic Workspace Manager の概要](#)
- [Analytic Workspace Manager を使用したスタンダード・フォーム・ワークスペースの作成](#)
- [事例 : Global アナリティック・ワークスペースの作成](#)
- [事例 : Sales History アナリティック・ワークスペースの作成](#)
- [集計データの生成](#)
- [事例 : GLOBAL アナリティック・ワークスペースのデータの集計](#)
- [アプリケーションに対するアナリティック・ワークスペースの有効化](#)
- [アナリティック・ワークスペースのデータのリフレッシュ](#)

アナリティック・ワークスペースの作成方法

アナリティック・ワークスペースの作成方法は、次の中から選択できます。

- **Analytic Workspace Manager**。アナリティック・ワークスペース作成ウィザードは、アナリティック・ワークスペースを作成する最も簡単な方法です。このウィザードについてはこの章で説明します。ウィザードでは DBMS_AWM 文の SQL スクリプトを作成し、その場で実行することも、後で実行するために保存することもできます。

- **Oracle Warehouse Builder。**Oracle Warehouse Builder を使用してソース・データをスター・スキーマに変換した場合、わずかな追加手順を実行するだけでアナリティック・ワークスペースを作成できます。
- **DBMS_AWM PL/SQL パッケージ。**DBMS_AWM パッケージを使用すると、スクリプト内の作成プロセスを自動化でき、バッチ・ウィンドウ内で夜間に行うようスケジュールできます。ほとんどの本番システムに、この方法が推奨されます。

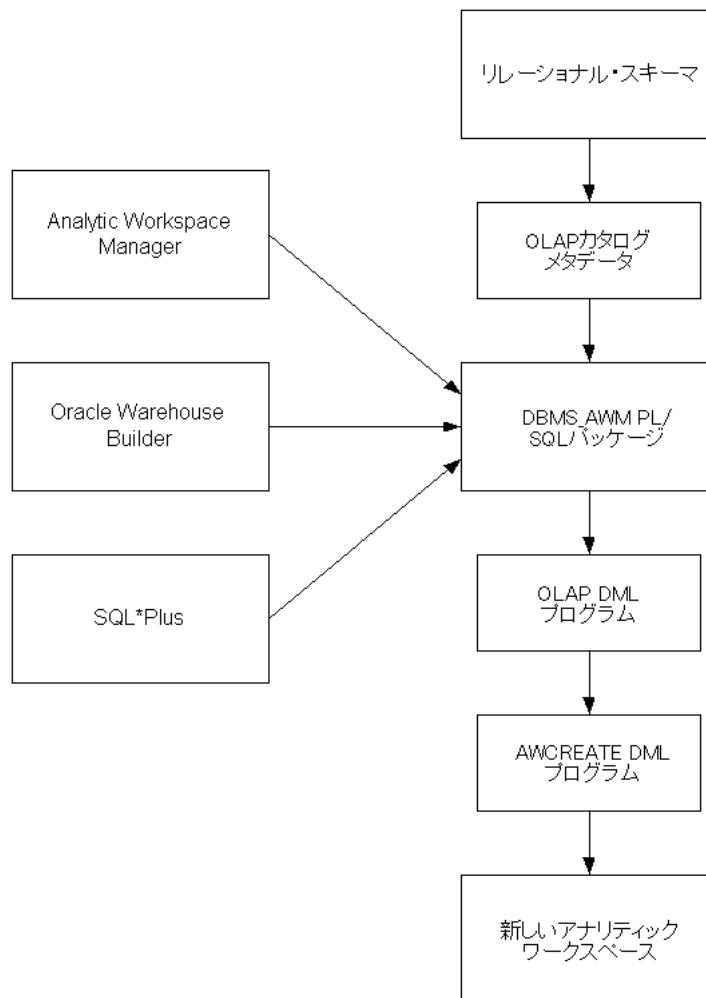
アナリティック・ワークスペース作成ウィザードを使用して基本スクリプトを作成した後も、編集して変更を加えることができます。これは、スクリプトをすべて手書きでコード化する際の単調さと手間を大幅に省くことができる方法です。アナリティック・ワークスペースの設計では、グラフィカルな方法よりも DBMS_AWM パッケージを使用する方が柔軟性があります。

- **OLAP DML プログラム。**各セッションにアタッチされる EXPRESS アナリティック・ワークスペースには、DBMS_AWM パッケージのプロシージャに厳密に対応するプログラムがいくつか含まれます。たとえば、DBMS_AWM.CREATE_AWCUBE プロシージャは、実行用にその引数を OLAP DML プログラムの CREATE_AWCUBE に渡します。OLAP DML が SQL より使いやすい場合、直接これらのプログラムをコールする OLAP DML プログラムを開発することもできます。

図 6-1 に、これらの方法の関係を示します。

こうした方法にかかわらず、設計では同じ決定を行い、最適に実行されるアナリティック・ワークスペースを生成する必要があります。これらの決定によって、データをいかに素早く取得できるかに影響するデータの格納方法が決まります。この章では、Analytic Workspace Manager を使用する状況における設計問題を説明しますが、別の方法を選択した場合でもこの情報が役立ちます。

図 6-1 アナリティック・ワークスペースの作成に使用されるコンポーネント



参照：

- OLAP Bridge を使用したアナリティック・ワークスペースの作成手順については、『Oracle Warehouse Builder ユーザーズ・ガイド』を参照してください。
- DBMS_AWM パッケージおよびこのパッケージによってコールされる OLAP DML プログラムの完全な詳細は、『Oracle OLAP リファレンス』を参照してください。

Analytic Workspace Manager の概要

Analytic Workspace Manager は、アナリティック・ワークスペースを作成、開発および管理するための主なツールです。Analytic Workspace Manager は、いくつかの PL/SQL パッケージおよび OLAP DML に GUI を提供する Java アプリケーションです。

コンソールまたはメイン・ウィンドウでは、OLAP カタログ・ビューおよびオブジェクト・ビューの 2 つのビューが提供されます。これら 2 つのビューを「View」メニューで切り替えることができます。さらに、これらのビューには、メニュー、ツールバー、ナビゲーション・ペインおよび表示ペインがあります。ナビゲーション・ペインでオブジェクトを選択すると、右側の表示ペインでそのオブジェクトの詳細情報が提供されます。オブジェクトを右クリックすると、そのオブジェクトに対する適切なアクションを選択できるメニュー項目が表示されます。

また、OLAP Worksheet を開くことによって、OLAP DML を使用した対話型セッションを行うこともできます。コンソールと OLAP Worksheet は同じセッションを共有するため、切り替えることでそれぞれのワークスペースの最新のビューを取得できます。

Analytic Workspace Manager は、状況依存ヘルプを含む完全なオンライン・ヘルプ・システムを持ちます。

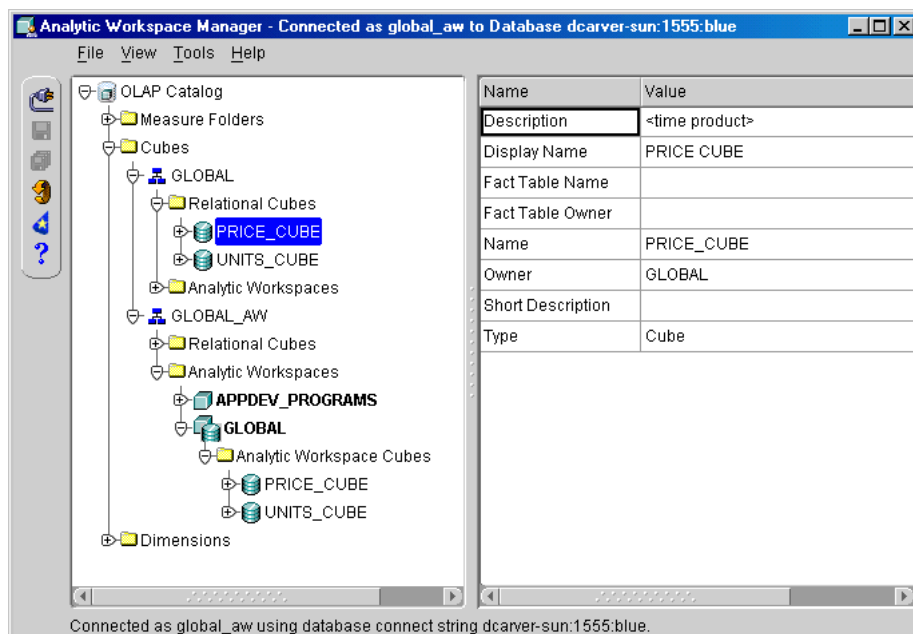
OLAP カタログ・ビュー

OLAP カタログ・ビューは、CWM1 および CWM2 の両方で、所有するか読み込みアクセス権を付与された OLAP カタログ・メタデータをすべて表示します。このビューでは、メジャー・フォルダ、キューブおよびディメンションなどの主要な論理オブジェクトが表示されます。ビューでメタデータの作成または変更はできません。これを行うには、Oracle Enterprise Manager、Oracle Warehouse Builder または CWM2 PL/SQL プロシージャを使用する必要があります。

アナリティック・ワークスペースを作成すると、OLAP カタログ・ビューには、リレーショナル・モデルの論理キューブに加え、アナリティック・ワークスペース内のインスタンス化されたキューブが表示されます。ワークスペース・キューブに対して集計プランを作成およびデプロイすることによって、メタデータを増やすことができます。集計プランはワークスペース・キューブに関連付けられ、アナリティック・ワークスペースに格納されます。集計プランによって、キューブ内のデータを解決するルールが提供されるので、すべてのレベルで値が計算されます。

図 6-2 に、OLAP カタログ・ビューを示します。ビューでは、GLOBAL スキーマの PRICE_CUBE というリレーショナル・キューブが選択されており、右側のペインにはそのキューブの情報が表示されています。GLOBAL で定義されているリレーショナル・キューブから GLOBAL_AW スキーマ内にアナリティック・ワークスペースが作成されていることが分かります。

図 6-2 Analytic Workspace Manager の OLAP カタログ・ビュー



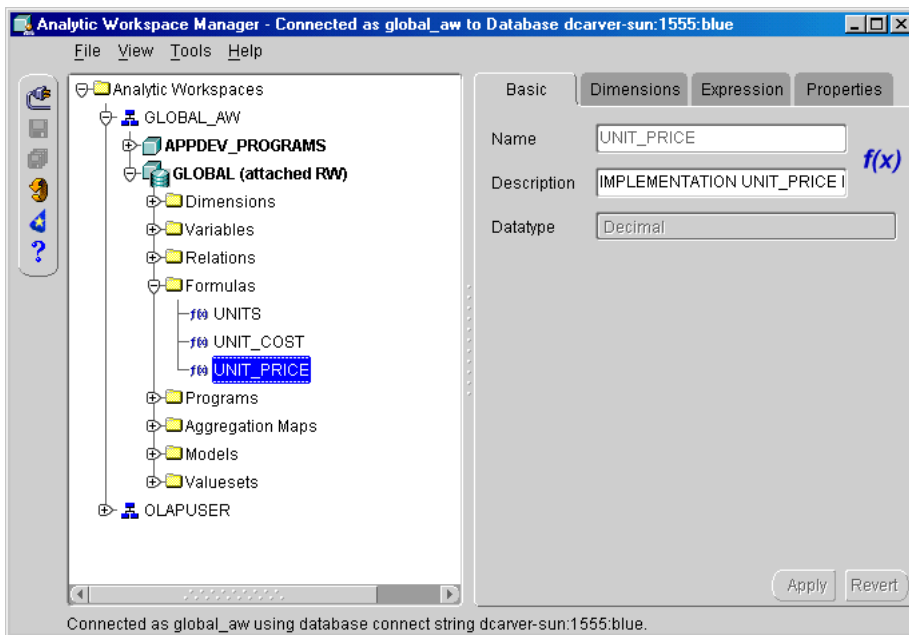
オブジェクト・ビュー

オブジェクト・ビューでは、OLAP DML に対する GUI を提供します。このビューでは、個々のワークスペース・オブジェクトを作成、変更および削除できます。オブジェクト・ビューで利用可能なツールを使用して、アナリティック・ワークスペース作成ウィザードで作成したワークスペースを変更できます。

オブジェクト・ビューは、ワークスペース・オブジェクトの定義の管理に主に役立ちます。ワークスペース・オブジェクトの内容を管理したりプログラムを実行する必要がある場合、または、OLAP DML の使用に習熟している場合は、OLAP Worksheet を開いて OLAP DML を完全に使用することができます。

図 6-3 に、オブジェクト・ビューを示します。UNIT_PRICE という名前の式が現在選択されており、右側のペインにはその式の情報が表示されています。

図 6-3 Analytic Workspace Manager のオブジェクト・ビュー

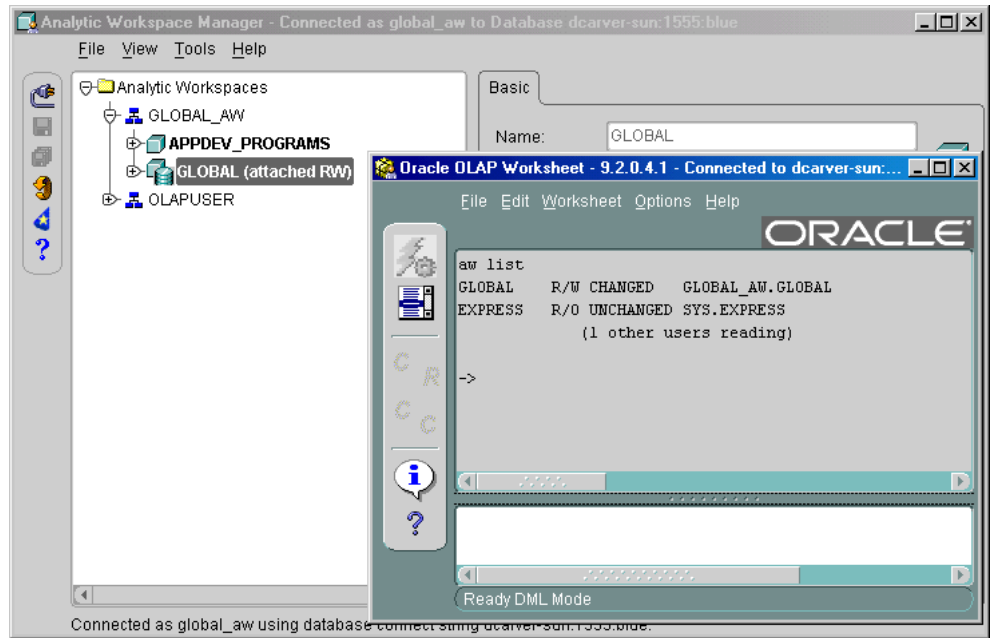


OLAP Worksheet

OLAP Worksheet は、Analytic Workspace Manager のコンソールとは別のウィンドウで開かれます。このウィンドウには、メニュー、ツールバー、OLAP DML コマンド用の入力ペイン（下部）および出力ペイン（上部）があります。

図 6-4 に、Analytic Workspace Manager から開いた OLAP Worksheet を示します。GLOBAL ワークスペースが、OLAP Worksheet と Analytic Workspace Manager の両方で、読み込み / 書き込みアクセス権にアタッチされている（それぞれ、AW LIST コマンドおよびオブジェクト・ビューで表示されている）ことが分かります。2つのアプリケーションで同じセッションを共有しているためです。

図 6-4 Analytic Workspace Manager から開いた OLAP Worksheet



Analytic Workspace Manager でデータベース接続を開く

データベースに接続するには次の手順を実行します。

1. 「File」メニューから「**Connect**」を選択します。
「Connect to Database」ダイアログ・ボックスが表示されます。
2. データベースのユーザー名、パスワードおよびサービスを入力します。次に「**OK**」を選択します。

host:port:sid の形式でサービスを指定します (myhost-sun:1521:rel10i など)。

Analytic Workspace Manager を使用したスタンダード・フォーム・ワークスペースの作成

アナリティック・ワークスペース作成ウィザードを使用して、OLAP カタログ内の 1 つまたは複数のキューブからワークスペースを作成することができます。結果として作成されるワークスペースは**データベース・スタンダード・フォーム**です (第 8 章を参照)。

アナリティック・ワークスペース作成ウィザードでは、最小限の決定でアナリティック・ワークスペースを作成できるよう、妥当なデフォルト値が提供されます。デフォルト設定を受け入れることによって、多くのデータ型に適切なデータの物理格納モデルを作成します。OLAP タイプの分析に習熟していない場合、最初はデフォルト設定でアナリティック・ワークスペースを作成し、使いながらパフォーマンス特性が容認できるかどうかを探る方がいいかもしれません。

ただし、本番システムでは良好なパフォーマンスが重要です。最適なパフォーマンスを発揮するワークスペースを作成するには、データの特性を調査し、拡張記憶域オプションを要件に対して適切に設定します。

アナリティック・ワークスペースのスキーマの選択

アナリティック・ワークスペースは、リレーショナル表とは別のスキーマに常に作成します。こうすることによって、単一のネームスペース内の一意名の定義における矛盾を回避できます。

拡張記憶域オプションの設定

記憶域設定は、パフォーマンスに多大な影響を与えることがあります。不適切に設定すると、データのロードおよび実行時の分析の両方でパフォーマンスが低下します。稠密な時間ディメンション、および、その他のディメンションにわたるランダムなスパース性（非 NULL 値が 30% 未満）など、予期した特性をデータが持つ場合、デフォルト設定が適切です。

コンポジット・ディメンションの定義

コンポジット・ディメンションは、スパースなデータの格納に必要な領域のサイズを減らす手段です。コンポジットがない場合、アナリティック・ワークスペースには、値が NULL (NA) の場合でも、ディメンション・メンバーの各組合せの値が格納されます。コンポジットがある場合、実際にデータを持つディメンション・メンバーの組合せのみが格納されます。データ値が存在するディメンション・メンバーの組合せを**タプル**と呼びます。変数にデータ値を追加すると、新しいコンポジット・タプルが自動的に作成されます。

コンポジット内にリストされているディメンションの順序によって、コンポジット・タプルの格納順が制御されます。ディメンションは通常、サイズ（つまり、メンバー数）によって順序付けられているので、最大のものが最初に変化し、最小のものが最後に変化します。最初に変化するディメンション・メンバーと一緒にクラスタ化され、最後に変化するディメンション・メンバーが最も広範囲に分散されるように、コンポジット・タプルは格納されます。コンポジットの定義では、最初のディメンションが最初に変化し、最後のディメンションが最後に変化します。

たとえば、<CUSTOMER PRODUCT CHANNEL> のコンポジットでは、CUSTOMER が最初に変化し、CHANNEL が最後に変化します。

Analytic Workspace Manager では、データがスパースである（つまり、キューブ内のセル数と比較してデータ値が少ない）ことを想定し、各キューブに 1 つのコンポジットを作成しま

す。コンポジットには、**Time** を除くすべてのディメンションが含まれます。メジャーを最初に **Time** ディメンションで定義して（サイズにかかわらず）、時間ベースの分析およびデータ・リフレッシュを行いやすくする場合があります。

Time ディメンションがスパースな場合は、**Time** を含むコンポジットを定義します。別に稠密なディメンションがある場合、そのディメンションを除くコンポジットを定義します。稠密でも非常に小さなディメンションの場合、コンポジット内の最後のディメンションとして含めることができます。最大のメンバー数を持つディメンションから最小メンバー数のディメンションの順に、コンポジット内に含めるようにしてください。

キューブ内のディメンションの順序付け

キューブにリストされているディメンションの順序は、データのディスクへの格納方法を決めるため、パフォーマンスに影響を与えます。キューブ内の最初のディメンションが最初に変化するディメンションで、最後のディメンションが最後に変化するディメンションです。キューブ内の各メジャーのデータは線形ストリームとして格納され、最初に変化するディメンションの値は一緒にクラスタ化されます。

データ記憶域は通常、ロードに対して最適化されます。コンポジットの前に稠密なディメンション（**Time** など）をリストします。稠密なディメンションが複数ある場合、最大のディメンションを最初にリストします。

セグメント・サイズの設定

セグメントとは、メジャーのデータを格納するために予約されている連続するディスク領域のサイズです。パフォーマンスは、メジャーのデータがすべて1つのセグメントに格納されている場合に最高です。

Analytic Workspace Manager では、最初のロードに対して十分な大きさのセグメントを作成します。この設定で、コンポジットの外にある **Time** ディメンションに対しては正確な大きさの、コンポジット・タプルが作成される際の格納用には十分な大きさのセグメントを定義します。データを更新する際は、最初のセグメントと同じサイズの2つ目のセグメントが作成されます。その後の更新分はこのセグメントに格納され、ディメンションの1つにおいて容量に到達すると、その時点で同じサイズの3つ目のセグメントが作成される、というように続きます。

セグメント・サイズを手動で設定すると、将来の増加分を見込んだセグメントを作成できます。最初に変化する稠密なディメンション（リストの最初にあるもの）に対しては特に、十分な領域を指定してください。ただし、予約しているディスク領域は他のユーザーが使用できない領域であることを踏まえておく必要があります。現実的な増加分をはるかに超えるサイズのセグメントは作成しないでください。

指定できるのは、多次元変数のセグメント・サイズのみです。1つのコンポジットまたは1つのディメンションによってのみディメンション化されている変数のセグメント・サイズは指定できません。

作成オプションの選択

アナリティック・ワークスペース作成ウィザードを使用すると、ロードするデータの量およびロードを開始する時期を決めることができます。

すべてのオブジェクト定義および次に示すデータの選択を使用して、アナリティック・ワークスペースを作成できます。

- データなし
- ディメンション表のすべてのデータ（ファクト表のデータはなし）
- ディメンション表およびファクト表のすべてのデータ

部分作成を使用すると、データをロードする前にオブジェクト定義に対して手動で変更を加えることができます。データを後でロードするには、アナリティック・ワークスペースのリフレッシュ・ウィザードを使用します。

スクリプトの生成

大量のデータを処理している場合は、スクリプトを生成すると特に有効です。スクリプトを使用すると次のことが実行できます。

- 作成プロセスに変更を加える
- 作成プロセスをバッチ・ウィンドウ内で夜間に実行するようスケジュールする

スタンダード・フォームのワークスペースを作成する基本手順

データベース・スタンダード・フォームのワークスペースを作成するには、次の手順に従います。

1. データベース・インスタンスを OLAP で使用できるように構成します。永続表領域、一時表領域および UNDO 表領域を定義し、データベース・パラメータをデータ・ロードに適切な値に設定します。詳細は、[第 12 章「Oracle OLAP の管理」](#)を参照してください。
2. アナリティック・ワークスペースを所有するユーザーを定義します。そのユーザーに OLAP_USER ロールおよびソース・データ表に対する SELECT 権限を付与します。

リレーショナル表と同じスキーマにワークスペースを作成できますが、そうすると、単一のネームスペース内に一意名を定義する際に問題が生じます。
3. データのスパース特性を調査します。Time ディメンションにわたって稠密で、その他のすべてのディメンションにわたってスパースなデータには、デフォルトのデータ記憶域設定が適切です。データにこうしたスパース特性がある場合、またはこのような特性を限定できない場合も、デフォルトのデータ記憶域設定を使用します。あるいは、アナリティック・ワークスペース作成ウィザードを実行する際に、データ・キューブに適切な **コンボジット** を定義します。

4. **Analytic Workspace Manager** を開き、この目的で以前に定義したユーザーとしてデータベース・インスタンスに接続します。
 5. ログ・ファイルを生成する場合、「Tools」メニューから「**Configuration**」を選択します。詳細な情報については、「**Help**」をクリックします。
 ログ・ファイルには、様々なウィザードで生成されたメッセージが格納されます。ログ・ファイルによって追加の情報が提供されることはありませんが、ウィザードが失敗し、その原因を調査する場合に役に立ちます。
 6. OLAP カタログ・ビューで、ソース・データのディメンション、階層、メジャーおよびキューブを定義しており、現行セッションからこれらの論理オブジェクトへのアクセス権を所有していることを確認します。
 7. 「Tools」メニューから、「**Create Analytic Workspace Using Wizard**」を選択します。このウィザードにおける手順を完了させます。コンポジットを定義する必要がある場合、拡張記憶域オプションを選択してください。
 各手順の具体的な情報を取得するには、「**Help**」ボタンをクリックします。
 8. 現時点または後で、**BI Beans** に対してワークスペースを有効化できます。アナリティック・ワークスペースを集計データおよびカスタム・メジャーで充実させるまで、有効化を延期できます。
 ワークスペースで他のタイプのアプリケーションをサポートする場合、適切なウィザードを使用して後で有効化できます。
 9. 「File」メニューから「**Save**」を選択します。このオプションは、現行のセッションでデータベースに対して行われたすべての変更をコミットします。
 10. データのすべてをロードせずにオブジェクトを定義する作成オプションを選択した場合、作成を完了する準備が整ったら、アナリティック・ワークスペースのリフレッシュ・ウィザードを実行します。
- 完了すると、リレーショナルなスター・スキーマまたはスノーフレーク・スキーマからフェッチされた詳細データが移入されたアナリティック・ワークスペースが作成されます。

事例 : Global アナリティック・ワークスペースの作成

次の事例では、GLOBAL スター・スキーマからアナリティック・ワークスペースを作成する際に行う選択について説明します。第 3 章に、スキーマの表を示しています。

GLOBAL_AW ワークスペース・ユーザーの定義

この例では、ソース表とは異なるスキーマに GLOBAL アナリティック・ワークスペースを作成します。例 6-1 に、Analytic Workspace Manager を使用し、GLOBAL スター・スキーマへアクセスするための十分なアクセス権を持った GLOBAL_AW ユーザーを定義する SQL コマンドを示します。あるいは、Oracle Enterprise Manager でユーザーを定義することもできます。

例 6-1 GLOBAL_AW ユーザーを定義する SQL スクリプト

```
CREATE USER "GLOBAL_AW" PROFILE "DEFAULT"
  IDENTIFIED BY "global_aw" DEFAULT TABLESPACE "GLOBAL"
  TEMPORARY TABLESPACE "OLAPTEMP"
  QUOTA UNLIMITED ON "GLOBAL"
  QUOTA UNLIMITED ON "OLAPTEMP"
  ACCOUNT UNLOCK;

GRANT OLAP_USER TO GLOBAL_AW;

GRANT SELECT ON global.channel_dim TO global_aw;
GRANT SELECT ON global.product_dim TO global_aw;
GRANT SELECT ON global.customer_dim TO global_aw;
GRANT SELECT ON global.time_dim TO global_aw;
GRANT SELECT ON global.units_history_fact TO global_aw;
GRANT SELECT ON global.price_and_cost_history_fact TO global_aw;
```

GLOBAL のスパース特性の調査

SQL の SELECT コマンドを COUNT および COUNT (DISTINCT) ファンクションとともに使用することによって、結果として作成される多次元キューブがアナリティック・ワークスペース内でどのくらい稠密となるかを見積ることができます。

PRICE_AND_COST_HISTORY_FACT 表は、考え得る 1728 のディメンション値の組合せ (48 か月 × 36 製品) のうち、UNIT_PRICE および UNIT_COST の両方に値を持つ 1407 の行を持ちます。Price キューブ (PRICE_AND_COST_HISTORY_FACT 表にマップされている) は 80% の稠密度なので、コンポジットによってパフォーマンスは向上せずに実際低下しています。稠密なキューブはかなりまれで、アナリティック・ワークスペース作成ウィザードや DBMS_AWM PL/SQL パッケージでもサポートされていません。したがって、データをロードする前にこのワークスペース・オブジェクト定義にいくつかの変更を加える必要があります。

UNITS_HISTORY_FACT 表は、考え得る 316,224 のディメンション値の組合せ (48 か月 × 36 製品 × 61 顧客 × 3 チャネル) のうち、UNITS に値を持つ 110,166 の行を持ちます。このキューブは稠密度 30% です。これは、コンポジットにとって十分スパースです。

Time は月レベルで集計されるので、Time ディメンションはおそらく稠密です。このキューブではデフォルトのコンポジットを使用できます。

アナリティック・ワークスペース作成ウィザードの実行

アナリティック・ワークスペース作成ウィザードの実行時には次のように選択します。

- 「Choose Data Loading Options」 ページで、次の手順を実行します。

- 2 つ目の作成オプション「**Build analytic workspace and load dimensions**」を選択します。これを選択すると、データをロードする前に変数定義に変更を加えることができます。
- 「**Generate unique keys**」ボックスを選択解除します。ディメンション表は、すべてのレベルにサロゲート・キーを使用して、ディメンション値の一意性を保証します。
- 「Choose Advanced Storage and Naming Options」ページでは、ワークスペース・オブジェクトの名前付けの際に接頭辞はありません。アナリティック・ワークスペースが、リレーショナル表とは別のスキーマに作成されているからです。キューブ名の接頭辞はオプションですが、アナリティック・ワークスペースに複数のキューブが含まれる際に役に立つ場合があります。

選択した作成オプションによって、アナリティック・ワークスペース作成ウィザードで、すべてのワークスペース・オブジェクトが定義され、データはすべてディメンション表からフェッチされます。メジャーは定義されますが、データを持ちません。データをロードする前にオブジェクト定義を変更できます。

(この例にかわる方法は、SQL スクリプトを作成して DBMS_AWM へのコールを変更し、アナリティック・ワークスペースが正しく生成され、変更の必要もないようにすることです。)

オブジェクト定義の手動による変更

UNIT_PRICE および UNIT_COST などのメジャー名は、変数の集計に使用される、アナリティック・ワークスペース内の式に与えられます。データ自体は、UNIT_PRICE_VARIABLE および UNIT_CUBE_VARIABLE などの名前の変数に格納されます。

OLAP Worksheet および OLAP DML を使用すると、UNIT_PRICE_VARIABLE および UNIT_COST_VARIABLE に次の 2 つの変更を加えることができます。

- **コンポジットの削除。** 6-12 ページの「[GLOBAL のスペース特性の調査](#)」で説明しているように、Price キューブは稠密度 80% です。UNIT_PRICE_VARIABLE および UNIT_CUBE_VARIABLE には、Price キューブのデータが格納されます。これらの変数は稠密なため、コンポジットでこれらを定義すると、実際にパフォーマンスが低下します。
- **データ型の変更。** Price キューブのデータは、SHORTDECIMAL データ型で円滑に処理されます。記憶域を保存し、パフォーマンスを最大限向上させるには、DECIMAL (8 バイト) および SHORTDECIMAL (4 バイト) を可能なかぎり使用します。DECIMAL は、デフォルトの数値データ型です。データ型の完全なリストについては、Analytic Workspace Manager の「Help」を検索してください。

次に、単なる変更ではなく、変数を再定義する必要がある大きな変更を示します。

次の手順を実行して変数を再定義します。

1. オブジェクト・ビューで、GLOBAL アナリティック・ワークスペースに読み込み / 書き込みアクセス権をアタッチします。
2. 「Tools」メニューから「**OLAP Worksheet**」を選択します。
3. 下部のペイン（問合せウィンドウ）で、次のコマンドを入力し、2 つの変数のオブジェクト定義を表示します。

```
FULLDSC price_cube_unit_price_variable price_cube_unit_cost_variable
```

4. 上部のペイン（応答ウィンドウ）から完全なオブジェクト定義を選択し、それらをテキスト・エディタに貼り付けます。
5. 定義を次のコード例のように編集します。冒頭の DELETE コマンド、DEFINE コマンドにおけるデータ型およびディメンションの変更、各 DEFINE コマンドの後に追加されている CONSIDER コマンド、および、1 行の完全なコマンドでリストされているスタンダード・フォームの PROPERTY コマンドに注目してください。

```
DELETE PRICE_CUBE_UNIT_PRICE_VARIABLE PRICE_CUBE_UNIT_COST_VARIABLE
```

```
DEFINE PRICE_CUBE_UNIT_PRICE_VARIABLE VARIABLE SHORTDECIMAL <TIME PRODUCT>
CONSIDER PRICE_CUBE_UNIT_PRICE_VARIABLE
PROPERTY 'AW$CLASS' 'EXTENSION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '20MAY03_10:37:08'
PROPERTY 'AW$PARENT_NAME' 'PRICE_CUBE_UNIT_PRICE'
PROPERTY 'AW$ROLE' 'VARIABLE'
PROPERTY 'AW$SEGWIDTH_CMD' 'chgdfn
GLOBAL_AW.GLOBAL!PRICE_CUBE_UNIT_PRICE_VARIABLE segwidth 120 72'
PROPERTY 'AW$STATE' 'CREATED'
```

```
DEFINE PRICE_CUBE_UNIT_COST_VARIABLE VARIABLE SHORTDECIMAL <TIME PRODUCT>
CONSIDER PRICE_CUBE_UNIT_COST_VARIABLE
PROPERTY 'AW$CLASS' 'EXTENSION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '20MAY03_10:37:08'
PROPERTY 'AW$PARENT_NAME' 'PRICE_CUBE_UNIT_COST'
PROPERTY 'AW$ROLE' 'VARIABLE'
PROPERTY 'AW$SEGWIDTH_CMD' 'chgdfn GLOBAL_AW.GLOBAL!PRICE_CUBE_UNIT_COST_VARIABLE
segwidth 85 72'
PROPERTY 'AW$STATE' 'CREATED'
```

6. このファイルを、データベース内でディレクトリ・オブジェクトとして定義されているディスク・ディレクトリに保存するか移動します。その後で、OLAP DML の INFILE コマンドを使用してファイルを実行します。INFILE は、SQL の @ コマンドに相当します。

```
INFILE 'directory_object/filename'
```

応答ウィンドウでエラーをチェックします。エラーを修正するには、ファイルを編集して再び実行します。

7. これらの変更を保存するには、問合せウィンドウに次のコマンドを入力して実行します。

```
UPDATE  
COMMIT
```

作成の完了

ワークスペース・オブジェクト定義への変更をすべて行ったら、リフレッシュ・ウィザードを実行してデータをロードします (6-29 ページの「[アナリティック・ワークスペースのデータのリフレッシュ](#)」を参照)。ディメンションはリフレッシュする必要はありません。キューブのみリフレッシュします。

ワークスペースは、この時点または後で、集計プランをデプロイして有効化できます。

事例 : Sales History アナリティック・ワークスペースの作成

Global はこのマニュアルの多くの例で使用されていますが、Sales History はデータ・セット特性が大きく異なり、これに相応して異なる作成の選択セットを示します。

Sales History (SH) は、OLAP カタログに格納されている完全に定義済の論理モデルに加えて、Oracle Database とともに提供されるサンプルのスター・スキーマです。SH スキーマは、SALES および COSTS の 2 つのキューブを持ちます。SALES キューブは 5 つのディメンションを持ち、COSTS キューブはこれらのディメンションのうち 2 つを使用します。この事例では、SALES キューブのみを使用します。

参照： Sales History の詳細は、『Oracle サンプル・スキーマ』を参照してください。

SH 作成用起動パラメータの定義

大きなアナリティック・ワークスペースを作成する際、どのくらい迅速に作成が進むかは、Oracle Database の起動パラメータに左右されます。例 6-2 に、Sales History を作成するための init.ora ファイルの設定をいくつか示します。これらの設定の詳細は、[第 12 章](#)を参照してください。

例 6-2 Sales History の作成用起動パラメータ

```
UNDO_TABLESPACE=OLAPUNDO  
UNDO_MANAGEMENT=AUTO  
PGA_AGGREGATE_TARGET=128M
```

SH の表領域の定義

GLOBAL アナリティック・ワークスペースが、その最大キューブ内のベース・レベルのデータに 100 万個未満のセルを持つのに対して、Sales History の COST キューブは、235 兆個以上のセルを持ちます。Sales History はサンプル・スキーマとしては非常に大きなサイズですが、本当のアプリケーションの平均よりは小さなサイズです。ただし、このスキーマの使用のためにリソースを明確に割り当てないと、このサイズによって作成が失敗する場合があります。作成では、十分な一時表領域および永続表領域が必要となります。

- Sales History アナリティック・ワークスペースで使用するための表領域を定義します。この表領域には、ベース・レベルのデータ、格納集計データ、予測データなどを保持するのに十分な大きさを定義します。可能な場合、個別の物理ディスク上の拡張ファイルを定義します。最高のパフォーマンスを得るためにも、スター・スキーマと同じ表領域を使用しないでください。
- SALES キューブのデータを保持するのに十分な大きさの一時表領域を定義します。EXTENT MANAGEMENT SIZE は、256KB などの小さな値を使用します。

例 6-3 に、Sales History の表領域をどのように定義するのかを示します。

例 6-3 Sales History アナリティック・ワークスペースの表領域の定義用 SQL スクリプト

```
/* Create a permanent tablespace on four disks */
CREATE TABLESPACE sh_aw DATAFILE '/disk1/oradata/sh_aw1.dbf' SIZE 64M AUTOEXTEND ON
NEXT 64M MAXSIZE 1024M EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

ALTER TABLESPACE sh_aw ADD DATAFILE '/disk2/oradata/sh_aw2.dbf' SIZE 64M REUSE
AUTOEXTEND ON NEXT 64M MAXSIZE 1024M, '/disk3/oradata/sh_aw3.dbf' SIZE 64M REUSE
AUTOEXTEND ON NEXT 64M MAXSIZE 1024M, '/disk4/oradata/sh_aw4.dbf' SIZE 64M REUSE
AUTOEXTEND ON NEXT 64M MAXSIZE UNLIMITED;

/* Create a temporary tablespace on four disks */
CREATE TEMPORARY TABLESPACE sh_temp TEMPFILE '/disk1/oradata/sh_aw.tmp' SIZE 64M
REUSE AUTOEXTEND ON NEXT 64M MAXSIZE 1024M EXTENT MANAGEMENT LOCAL UNIFORM SIZE
256K;

ALTER TABLESPACE sh_temp ADD TEMPFILE '/disk2/oradata/sh_aw2.tmp' SIZE 64M REUSE
AUTOEXTEND ON NEXT 64M MAXSIZE 1024M, '/disk3/oradata/sh_aw3.tmp' SIZE 64M REUSE
AUTOEXTEND ON NEXT 64M MAXSIZE 1024M, '/disk4/oradata/sh_aw4.tmp' SIZE 64M REUSE
AUTOEXTEND ON NEXT 64M MAXSIZE UNLIMITED;
```

SH データのスパース特性の調査

SH リレーショナル・スキーマ内のデータは非常にスパースです。ディメンション・キーの多くは SALES ファクト表では外部キーとして使用されず、他の 4 つのディメンションの考え得るすべての組合せではほとんど使用されていません。たとえば、CUSTOMERS.CUST_ID

は 5100 個の値を持ちますが、そのうち、SALES.CUST_ID 列で使用されているのは 2557 個のみです。

Time もスパース・ディメンションで、使用されているディメンション・メンバーは、1826 個のうち 1075 個のみです。したがって、TIMES_DIM をコンポジットに含める必要があります。拡張記憶域オプションを選択することによって、5 つのすべてのディメンションを持つコンポジットを定義できます。TIMES は、PRODUCTS および CUSTOMERS より小さくても、コンポジット内の最初の（最初に変化する）ディメンションとしてリストし、時間ベースの分析およびデータ・メンテナンスを行いやすくします。他のディメンションは、大きい方から小さい方の順序でリストします。この情報は、ディメンション表に対して SELECT COUNT(*) を発行すると簡単に取得できます。

SH 作成の管理

SH のサイズは大きいので、作成における次の側面を管理する場合があります。

- 時間 : オフピーク時に作成を実行します。これを行うには、直接作成せずに作成用の SQL スクリプトを生成します。
- 進捗状況の監視 : 進捗状況を監視できるように SQL スクリプトにコメントを追加します。なんらかの理由で作成が失敗した場合や、作成を中断する必要がある場合、作成が停止された箇所からスクリプトを再開できます。
- 永続表領域のサイズ : 可能な場合、小さなデータ型でメジャーを定義します (6-13 ページの「[オブジェクト定義の手動による変更](#)」を参照)。このタイプの変更では、部分作成オプションの 1 つを選択する必要があります。

データ型の説明については、Analytic Workspace Manager の「Help」を検索してください。

アナリティック・ワークスペース作成ウィザードの実行

前の説明に基づいて Sales History を作成するには、アナリティック・ワークスペース作成ウィザードで次のように選択します。

- 「Choose Data Loading Options」ページで、次の手順を実行します。
 - 2 つ目の作成オプション「**Build analytic workspace and load dimensions**」を選択します。これを選択すると、データをロードする前に変数定義に変更を加えることができます。
 - 「**Generate unique keys**」ボックスを選択解除します。ディメンション表は、すべてのレベルにサロゲート・キーを使用して、ディメンション値の一意性を保証します。
- 「Choose Advanced Storage and Naming Options」ページで、「**Advanced Storage Options**」を選択します。これを選択すると、TIMES ディメンションを含むコンポジットを定義できます。

アナリティック・ワークスペースがリレーショナル表とは異なるスキーマに作成されているため、接頭辞は必要ありません。キューブ名の接頭辞はオプションですが、アナリティック・ワークスペースに複数のキューブが含まれる際に役に立つ場合があります。

- コンポジットを作成する際は、ディメンションすべてを含めて、次の順序で配置します。セグメント・サイズは指定しないでください。非常に大きなセグメントがコンポジットに自動的に割り当てられるためです。

TIMES
PRODUCTS
CUSTOMERS
PROMOTIONS
CHANNELS

- 「Choose Create and Enablement Options」 ページで、「**Save Script to File**」を選択します。この時点でワークスペースを有効化する必要はありません。

Sales History アナリティック・ワークスペースの作成

次の手順を実行して、Sales History アナリティック・ワークスペースを作成します。

1. 作成スクリプトを実行します。このスクリプトが正常に実行されたら、ディメンション表からのディメンション、階層、レベルおよび属性のすべてを持つアナリティック・ワークスペースが作成されます。
2. オブジェクト定義に対して任意の変更または追加を行います。
3. リフレッシュ・ウィザードを実行してデータをロードします (6-29 ページの「[アナリティック・ワークスペースのデータのリフレッシュ](#)」を参照)。ディメンションはリフレッシュする必要はありません。キューブのみリフレッシュします。

ワークスペースは、この時点または後で、集計プランをデプロイして有効化できます。

集計データの生成

アナリティック・ワークスペースには、最初はリレーショナル・スキーマからの詳細データのみが含まれます。ただし、データの集計に必要な階層、レベルおよび親リレーションも含まれます。アナリティック・ワークスペースの集計データは、マテリアライズド・ビューを使用するかわりになります。集計データはすべてアナリティック・ワークスペースに作成されるからです。実行時のパフォーマンスを最適化するには、いくつかの集計データを生成および格納する必要があります。

集計を計算するための方法

アナリティック・ワークスペース内のデータ・キューブは、次の 2 つの時点で解決できます。

- **必要に応じて実行時に行う。**集計値のセルは、問合せで集計値が要求されるまでは NA (NULL) です。その後、問合せに応じて集計が計算されます。このタイプの集計は、**即時集計**、または**実行時集計**と呼ばれます。実行時集計では、データは単に取得されるのではなく計算も必要になるため問合せ時間は長くなりますが、永続表領域に集計値用の記憶域は必要ありません。
- **データ・メンテナンス処理として行う。**DBA が集計値を計算し、それらをすべてのユーザーが共有できるようにアナリティック・ワークスペースに格納します。このタイプの集計データは、**事前計算集計**または**格納集計**と呼ぶ場合もあります。格納集計では問合せ時間は最短となりますが、アナリティック・ワークスペースのサイズが増加するため、リレーショナル・データベースのサイズも増加します。格納データの量は、データ・メンテナンスに使用できる時間（通常はバッチ・ウィンドウ）によって制限される場合もあります。キューブの完全な具体化は、バッチ・ウィンドウで許容されるよりも実際に時間がかかる場合があります。

ディメンションが複数の階層を持つか、階層が多くのレベルを持つ場合、メジャーを完全に集計すると、アナリティック・ワークスペースのサイズ（つまりデータベースのサイズ）が幾何学的に増加します。他方で、中間レベルのデータの多くはあまりアクセスされないか、ほとんどアクセスされない場合があります。

代表的な方法としては、データの一部をデータ・メンテナンス処理として集計し、残りのデータをオンデマンドで集計します。この方法は、**スキップ・レベル集計**と呼ばれます。データ・キューブは、記憶域から取得された値と問合せに対して計算された値との違いが分からない状態、および完全に解決された状態でアプリケーションに表示されます。スキップ・レベル集計が正常に実行されると、未解決のレベルを計算する時間は無視できます。

事前集計および格納するレベルの選択方法

事前集計のレベルを識別する優れた方法の 1 つは、各レベルでディメンション・メンバーの比率を決め、メンバーの比率を即時にロールアップさせ約 10:1 で保つことです。この比率によって、すべての回答が素早く戻されることが保証されます。データはアナリティック・ワークスペースに格納されるか、一度に 10 個の値を各集計値にロールアップすることによって計算されます。

この 10:1 の規則が最も適合するのは一部の判断に対してです。ほとんどアクセスされないことがないと分かっているレベルには高い比率を設定しておきたい場合もあります。あるいは、頻繁に使用されると分かっている場合は、低い比率でレベルを事前計算したい場合もあります。

集計プランについて

アナリティック・ワークスペース作成ウィザードで作成したアナリティック・ワークスペースは、各キューブに対して**集計プラン**と呼ばれるデフォルトの規則を持ちます。デフォルトのプランでは次のことが指定されています。

- 集計レベルは格納されない。すべての集計データは、問合せに回答を戻す必要に応じて実行時に計算されます。

- キューブ内のすべてのメジャーはデフォルトのプランを使用する。

デフォルトの集計プランによって、キューブ内のメジャーが常に完全に解決したデータをアプリケーションに表示することが保証されます。つまり、回答ではすべての階層のすべてのレベルが移入されています。ただし、デフォルトのプランでは、すべてのレベルがユーザー・セッション期間に計算される必要があることを指定しているので、これらのプランを使用すると、通常はパフォーマンスが受け入れがたいほど低下します。

そこで、データの一部を事前計算する集計プランを定義およびデプロイすることができます。各メジャーが自身の集計プランを持つか、キューブ内の任意数のメジャーが同じ集計プランを共有できます。作成する各集計プランでは、次のことを指定する必要があります。

- どの集計レベルを格納するか。
- キューブ内のどのメジャーがこのプランを使用するか。

集計プランはデプロイされるまで効力を持ちません。デプロイによって、アナリティック・ワークスペースのオブジェクトが作成および変更され、集計プランがサポートされます。そしてすべての格納集計レベルが計算されます。集計プランは数分で作成できますが、デプロイには、計算が必要なデータの量および利用可能なリソース次第でかなりの時間がかかる場合があります。オフピーク時の集計をスケジュールする必要がある場合もあります。

集計プランの作成およびデプロイ方法

集計プランは、キューブ内のすべてのメジャーまたは選択したメジャーに対してのみ使用できます。

集計プランの作成

集計プランを作成するには、次の手順を実行します。

1. スキーマ内のアナリティック・ワークスペースの名前が表示されるように、OLAP カタログ・ビューの「Cubes」フォルダを開きます。
2. ワークスペースの名前を右クリックします。
3. 「Create Aggregation Plan Using Wizard」を選択します。このウィザードにおける手順を完了させます。

各手順の具体的な情報を取得するには、「Help」ボタンをクリックします。

集計プランは、明示的に削除するまでアナリティック・ワークスペースの永続的な部分となります。

集計演算子の変更

集計プラン・ウィザードは常に SUM を集計の方法として指定します。ただし、この方法は次の演算子の 1 つに変更できます。

AVERAGE

FIRST
LAST
MAX
MIN

DBMS_AWM パッケージのプロシージャによって、既存の集計プラン内の演算子を変更されます。そのプロシージャ・コールの構文は次のとおりです。

```
EXECUTE DBMS_AWM.SET_AWCUBEAGG_SPEC_AGGOP('aggplan', 'aw_owner', 'aw_name', 'cube',  
'measure', 'dimension', 'operator')
```

OLAP API にこの変更を認知させるために、アナリティック・ワークスペースを再有効化するか、手動で CWM2_OLAP_METADATA_REFRESH.MR_AC_REFRESH プロシージャを実行する必要があります。

6-23 ページの「[Global Price キューブの集計](#)」の例を参照してください。構文の詳細は、『Oracle OLAP リファレンス』を参照してください。

集計プランのデプロイ

デプロイによって、最初に以前の集計データがすべて削除され、次に指定したレベルのデータが解決されます。

集計プランをデプロイするには、次の手順を実行します。

1. ワークスペースの集計プランの名前が表示されるように、OLAP カタログ・ビューを十分に開きます。
2. 集計プランの名前を右クリックし、「**Deploy Aggregation Plan Using Wizard**」を選択します。このウィザードにおける手順を完了させます。

集計プランは編集できますが、変更したプランを再デプロイするまでデータは変更されません。同様に、集計プランは削除できますが、同じメジャーに別のプランをデプロイするまで変更は行われません。

データをリフレッシュするたびに、集計プランを再デプロイする必要があります。詳細は、6-29 ページの「[アナリティック・ワークスペースのデータのリフレッシュ](#)」を参照してください。

事例 : GLOBAL アナリティック・ワークスペースのデータの集計

GLOBAL には 2 つのキューブが含まれます。Units キューブの集計は合計（デフォルト）しますが、Price キューブの集計は平均します。各キューブ内のすべてのメジャーは、キューブと同じ方法で集計します。

事前計算のレベルの識別

事前計算するレベルを識別するには、各レベルのディメンション・メンバーの数を知る必要があります。この情報は、SQL 文または OLAP DML コマンドを使用することによって、簡単に取得できます。

たとえば、SQL 文では次のように指定します。

```
SELECT COUNT(DISTINCT year_id) FROM global.time_dim;
```

また、OLAP DML コマンドでは、GLOBAL アナリティック・ワークスペースで次のように指定します。

```
SHOW NUMLINES (LIMIT(time TO time_levelrel EQ 'Year'))
```

どちらも、Year レベルの TIME ディメンション・メンバーの数を戻します。

Global は非常に小さなデータ・セットのため、隣接したレベルでディメンション・メンバーの 10:1 比率を持つものはほとんどありません。表 6-1 に、計算され、アナリティック・ワークスペースに格納されるレベルを示します。

表 6-1 Global ワークスペース内の事前計算レベル

ディメンション	レベル	メンバー	事前計算
TIME	Month	60	X
TIME	Quarter	20	--
TIME	Year	5	X
CUSTOMER	Ship_To	61	X
CUSTOMER	Account	24	--
CUSTOMER	Market_Segment	5	X
CUSTOMER	Total_Market	1	--
CUSTOMER	Warehouse	11	--
CUSTOMER	Region	3	X
CUSTOMER	All_Customers	1	--
PRODUCT	Item	36	X
PRODUCT	Family	9	--
PRODUCT	Class	2	X
PRODUCT	Total_Product	1	--

表 6-1 Global ワークスペース内の事前計算レベル

ディメンション	レベル	メンバー	事前計算
CHANNEL	Channel	3	X
CHANNEL	All_Channels	1	--

Global Price キューブの集計

次の手順を実行して、Price キューブを集計します。

1. 集計プランの作成ウィザードを実行して、PRICE_CUBE のプランを作成します。
2. オブジェクト・ビューで、読み込み / 書き込みモードで GLOBAL アナリティック・ワークスペースをアタッチします。
3. 「Tools」メニューから「OLAP Worksheet」を選択します。
4. OLAP Worksheet の「Options」メニューから「SQL Mode」を選択します。
5. 問合せウィンドウに次の SQL 文を入力します。

```
EXECUTE DEMS_AWM.SET_AWCUBEAGG_SPEC_AGGOP('price_aggplan', 'global_aw',
'global', 'price_cube', 'unit_price', 'time', 'AVERAGE')
```

```
EXECUTE DEMS_AWM.SET_AWCUBEAGG_SPEC_AGGOP('price_aggplan', 'global_aw',
'global', 'price_cube', 'unit_cost', 'time', 'AVERAGE')
```

6. OLAP カタログ・ビューで、集計プランのデプロイ・ウィザードを実行し、集計データを生成します。
7. 「File」メニューから「Save」を選択します。
8. GLOBAL アナリティック・スペースを OLAP API で使用できるようにします。

アプリケーションに対するアナリティック・ワークスペースの有効化

Oracle アプリケーションは通常、Oracle Database 内のリレーショナル表に対して実行されるよう設計されています。リレーショナル表は、アプリケーションによって設定されている一定の基準に準拠する必要があるため、メタデータの一部の形式がアプリケーションに対するデータの識別に使用されます。たとえば、BI Beans では、解決済データの埋込み合計ディメンション・ビューを持つスター・スキーマやスノーフレーク・スキーマ、および、スキーマを記述するための OLAP カタログ・メタデータが必要とされます。

同じアプリケーションであれば、変更を加えることなく、そのアプリケーションの使用が有効化されているアナリティック・ワークスペースに対して実行することができます。アナリティック・ワークスペースを有効化するには、次のことを実行している必要があります。

- アプリケーションで通常使用されているリレーショナル表と同じ基準に準拠する、アナリティック・ワークスペース・データのビューの作成。これらのビューでは OLAP_TABLE ファンクションを使用してワークスペース・データを抽出します。
- ビューへアクセスするためにアプリケーションに必要なすべてのメタデータの作成。

アナリティック・ワークスペースを有効化する方法

アナリティック・ワークスペースを有効化するには、次の手順を実行します。

1. スキーマのワークスペースが表示されるように、OLAP カタログ・ビューを十分に開きます。
2. 有効化するアナリティック・ワークスペースの名前を右クリックします。
3. 「Enable Workspace for OLAP API & BI Beans」

または

「Enable Workspace for Oracle9iAS Discoverer Using Wizard」を選択します。

このウィザードにおける手順を完了させます。各手順の具体的な情報を取得するには、「Help」ボタンをクリックします。

BI Beans に対する有効化について

OLAP API および BI Beans に対してアナリティック・ワークスペースを有効化する際は、スター・スキーマを形成するいくつかのビューを作成します。さらに、これらのビューで BI Beans アプリケーションにアクセスできるようにする CWM2 メタデータを作成します。

参照： ビューおよび CWM2 読み込み API の詳細は、『Oracle OLAP リファレンス』を参照してください。

ビューのスター・スキーマ

Beans が有効化されたアナリティック・ワークスペースのスター・スキーマには次のものが含まれます。

- 各階層のディメンション・ビュー
- ディメンション階層の各組合せのファクト・ビュー

これらのビューは、**埋込み合計**ビューと呼ばれる場合もあります。すべてのレベルのディメンション・メンバーが 1 つの列にリストされているためです。特定のメンバーが属するレベルおよびメンバー間の親子関係に関する情報は、別の列に格納されます。ファクト表内では、サマリー・データはベース・レベルのデータと混在するか、ベース・レベルのデータに埋め込まれます。これは、ファクト表にサマリー・データがなく、各レベルがディメンション表内の自身の列で表されるソースのスター・スキーマとは明らかに異なります。

表 6-2 に、GLOBAL アナリティック・ワークスペースを有効化する際に GLOBAL_AW スキーマに作成されるビューを示します。

表 6-2 BI Beans に対する GLOBAL アナリティック・ワークスペースのビュー

ビューの名前	説明
GLOB_GLOBA_CHANN_CHANN5VIEW	CHANNEL ディメンション・ビュー、CHANNEL_ROLLUP 階層
GLOB_GLOBA_CUSTO_MARKE6VIEW	CUSTOMER ディメンション・ビュー、MARKET_SEGMENT 階層
GLOB_GLOBA_CUSTO_SHIPM7VIEW	CUSTOMER ディメンション・ビュー、SHIPMENTS 階層
GLOB_GLOBA_PRODU_PRODU1VIEW	PRODUCT ディメンション・ビュー、PRODUCT_ROLLUP 階層
GLOB_GLOBA_TIME_CALEN2VIEW	TIME ディメンション・ビュー、CALENDAR 階層
GLOB_GLOBA_PRICE_CU4VIEW	PRICE_CUBE メジャー・ビュー
GLOB_GLOBA_SALES_CU10VIEW	SALES_CUBE メジャー・ビュー、CUSTOMER MARKET_SEGMENT 階層
GLOB_GLOBA_SALES_CU9VIEW	SALES_CUBE メジャー・ビュー、CUSTOMER SHIPMENTS 階層
GLOB_GLOBA_UNITS_CU12VIEW	UNITS_CUBE メジャー・ビュー、CUSTOMER MARKET_SEGMENT 階層
GLOB_GLOBA_UNITS_CU13VIEW	UNITS_CUBE メジャー・ビュー、CUSTOMER SHIPMENTS 階層

アナリティック・ワークスペースの OLAP カタログ・メタデータ

アナリティック・ワークスペースを BI Beans に対して有効化すると、前に説明したように OLAP カタログ・メタデータがリレーショナル・ビューに対して作成されます。

アナリティック・ワークスペースの OLAP カタログ・メタデータは表のセットに格納され、これは OLAPSYS が所有するビューのセットを介してアクセスできます。これらのビューのパブリック・シノニムには、ALL_OLAP2 の接頭辞が付きます。

BI Beans では、パブリック・シノニム MRV_OLAP2 を持つ特別なビューのセット（OLAPSYS が所有する）を問い合わせます。これらのビューは、パフォーマンスを向上させる特殊な構造を持った CWM2 メタデータ表から作成されます。

Analytic Workspace Manager の OLAP カタログ・ビューでメタデータを表示できます。あるいは、SQL コマンドを使用してビューを問い合わせることができます。例 6-4 に問合せの結果を示します。

例 6-4 CWM2 読み込み API の問合せ

```
SELECT * FROM OLAPSYS.MRV_OLAP2_CATALOGS;
```

CATALOG_ID	CATALOG_NAME	PARENT_CATALOG_ID	DESCRIPTION
1849	GLOBAL_CAT		Global Business Areas
475	XADEMO_MULTIKEY_CAT		XADEMO MULTIKEY Measures
669	XADEMO_CAT XADEMO		CWM Business Area

Oracle Discoverer に対してアナリティック・ワークスペースを有効化する方法

アナリティック・ワークスペースを有効化するには、次の手順を実行します。

1. OLAP カタログ・ビューを開いて、スキーマのワークスペースを表示します。
2. 有効化するアナリティック・ワークスペースの名前を右クリックします。
3. 「**Enable Workspace for OracleAS Discoverer Using Wizard**」を選択します。このウィザードにおける手順を完了させます。
各手順の具体的な情報を取得するには、「**Help**」ボタンをクリックします。
4. 必要に応じて最初に変更を保存し、アナリティック・ワークスペースをデタッチします。
5. ビューを作成するには、ウィザードで生成された SQL スクリプトを実行します。
6. End User Layer (EUL) を作成するには、Oracle Discoverer Administrator を使用して、ウィザードで生成された EEX ファイルをインポートします。

Oracle Discoverer に対する有効化について

Discoverer の有効化ウィザードでは、Analytic Workspace Manager を実行しているローカル・コンピュータ上に次の 2 つのファイルを生成します。

- Discoverer で要求される形式でワークスペース・データのビューを作成する SQL スクリプト
- End User Layer を作成するための XML を含む EEX ファイル

このスクリプトを実行し、EEX ファイルをインポートするまでは、アナリティック・ワークスペースは有効化されません。

このリリースでは、各ディメンションに対して 1 つの階層のみサポートされます。ディメンションに複数の階層がある場合、Discoverer でアクセスする階層を 1 つ選択する必要があります。

Discoverer に作成されるビュー

Discoverer に対してアナリティック・ワークスペースを有効化すると、2 つのビューのセットが生成されます。

1 つ目のビューのセットには、実際にデータを抽出する OLAP_TABLE ファンクション・コールが含まれます。このセットは、定義する必要のあるオブジェクト型および表型の数を抑えて、Discoverer で使用される形式のビュー（2 つ目のビューのセット）を簡単に作成できるようにするためにあります。このセットには次の 2 つのタイプのビューが含まれます。どちらのセットも、スキーマ、ディメンションおよびレベルの識別に、名前ではなく、文字識別子および数識別子を使用します。

- すべてのディメンションとメジャーを持つ完全なアナリティック・ワークスペースのビュー。この名前は次の形式となります。

`workspace_schema_FULL_VIEW`

`schema` には、S1 などのように S の後に数字を続けます。

- 各ディメンションのビュー。この名前は次の形式となります。

`TFDV_dimension_VIEW`

`dimension` には、D5 などのように D の後に数字を続けます。

2 つ目のビューのセットは 1 つ目のセットからデータを選択し、Discoverer で必要とされる形式でアナリティック・ワークスペース・データを表示します。このセットにも 2 つのタイプのビューがあります。

- ディメンション・レベルの各組合せのメジャー・ビュー。このビューは、前に説明した FULL_VIEW に対して実行されます。この名前は次の形式となります。

`FACTVIEW_schema_level_level_level...`

`schema` には、S1 などのように S の後に数字を続け、`level` には、L1 などのように L の後に数字を続けます。

- 各ディメンションのビュー。このビューは、前に説明した TFDV ビューに対して実行されます。この名前は次の形式となります。

`DV_dimension`

`dimension` には、D1 などのように D の後に数字を続けます。

図 6-5 に、ビュー間の関係を示します。

図 6-5 Discoverer のビュー間の関係

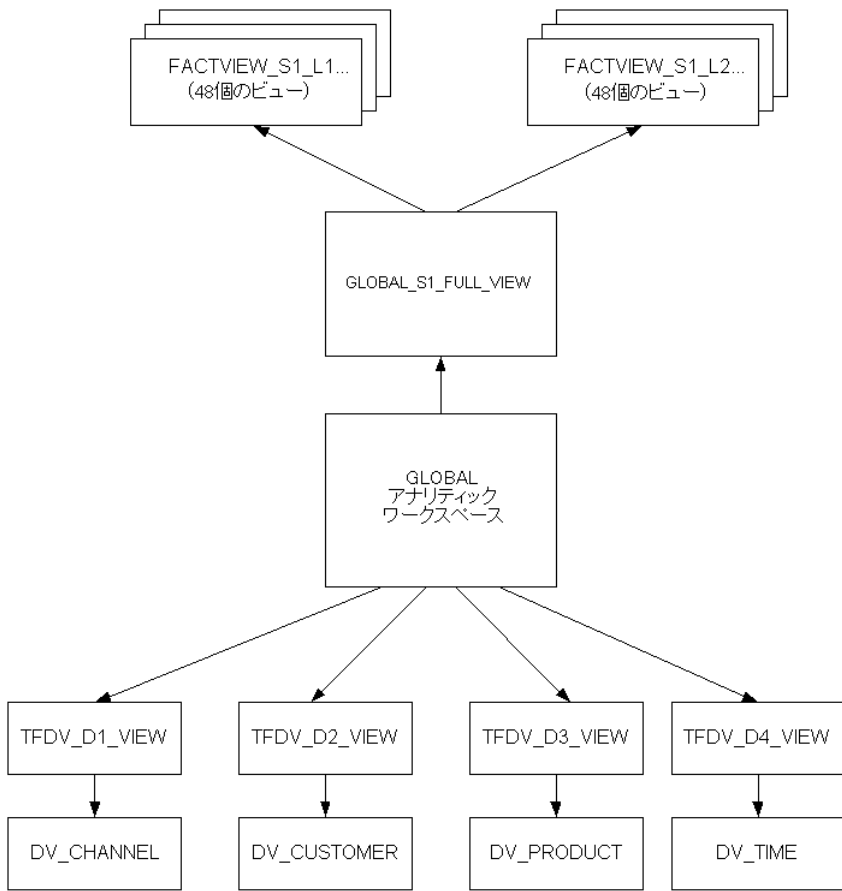


表 6-3 で、GLOBAL アナリティック・ワークスペースのビューについて説明します。

表 6-3 Oracle Discoverer に対する GLOBAL アナリティック・ワークスペースのビュー

ビュー	説明
GLOBAL_S1_FULL_VIEW	OLAP_TABLE ファンクションを使用して、GLOBAL アナリティック・ワークスペース内のすべてのディメンションおよびメジャーの完全なビューを作成します。
TFVD_D1_VIEW	OLAP_TABLE ファンクションを使用して、PRODUCT ディメンション・ビューを作成します。

表 6-3 Oracle Discoverer に対する GLOBAL アナリティック・ワークスペースのビュー

ビュー	説明
TFVD_D2_VIEW	OLAP_TABLE ファンクションを使用して、TIME ディメンション・ビューを作成します。
TFVD_D3_VIEW	OLAP_TABLE ファンクションを使用して、CHANNEL ディメンション・ビューを作成します。
TFVD_D4_VIEW	OLAP_TABLE ファンクションを使用して、CUSTOMER ディメンション・ビューを作成します。
FACTVIEW_S1_L1_L6_L10_L14	GLOBAL_S1_FULL_VIEW から、ALL_CHANNELS、ALL_CUSTOMERS、TOTAL_PRODUCT および YEAR レベルのすべてのメジャーを選択します。
FACTVIEW_S1_L1_L6_L10_L15	GLOBAL_S1_FULL_VIEW から、ALL_CHANNELS、ALL_CUSTOMERS、TOTAL_PRODUCT および QUARTER レベルのすべてのメジャーを選択します。
.	.
.	.
.	.
FACTVIEW_S1_L2_L9_L13_L15	GLOBAL_S1_FULL_VIEW から、CHANNEL、SHIP_TO、ITEM および QUARTER レベルのすべてのメジャーを選択します。
FACTVIEW_S1_L2_L9_L13_L16	GLOBAL_S1_FULL_VIEW から、CHANNEL、SHIP_TO、ITEM および MONTH レベルのすべてのメジャーを選択します。
DV_CHANNEL	TFVD_D3_VIEW から、CHANNEL ディメンション・ビューを選択します。
DV_CUSTOMER	TFVD_D4_VIEW から、CUSTOMER ディメンション・ビューを選択します。
DV_PRODUCT	TFVD_D1_VIEW から、PRODUCT ディメンション・ビューを選択します。
DV_TIME	TFVD_D2_VIEW から、TIME ディメンション・ビューを選択します。

アナリティック・ワークスペースのデータのリフレッシュ

一部の作成オプションではデータがロードされないので、アナリティック・ワークスペースを使用するには、初期リフレッシュを実行する必要があります。時間が経過すれば、すべて

のアナリティック・ワークスペースは新しいデータでリフレッシュする必要があります。データソースは他の新しいディメンション・メンバー同様、新しい期間を持ちます。

リフレッシュ・ウィザードの使用

アナリティック・ワークスペースのリフレッシュ・ウィザードで、選択したディメンションに新しいメンバーを追加し、選択したメジャーにデータをすべて再ロードします。ウィザードでは、新しいデータが元のデータと同じ表内にあることが要求されます。

データをリフレッシュするには、次の手順を実行します。

1. OLAP カタログ・ビューを開いて、スキーマのワークスペースを表示します。
2. 有効化するアナリティック・ワークスペースの名前を右クリックします。
3. 「**Refresh Analytic Workspace Using Wizard**」を選択します。このウィザードにおける手順を完了させます。ディメンションまたはメジャーを個々にリフレッシュしたり、両方をリフレッシュすることができます。
各手順の具体的な情報を取得するには、「**Help**」ボタンをクリックします。
4. 必要な場合、キューブを再有効化します。
5. 集計プランを再デプロイします。

異なるリレーショナル表からのリフレッシュ

リフレッシュ・ウィザードでは、アナリティック・ワークスペース作成ウィザードで最初に使用した同じ表にアクセスします。ただし、新しいデータを別の表のリレーショナル・スキーマにロードする場合があります。DBMS_AWM パッケージを使用して SQL でロード・プログラムを記述するか、次の手順に従います。

キューブをリフレッシュする基本的な手順は次のとおりです。

1. **Analytic Workspace Manager** のオブジェクト・ビューで、アナリティック・ワークスペースを読み込み / 書き込みモードでアタッチします。
2. オブジェクト・ビューで、「**Programs**」フォルダを開き、アナリティック・ワークスペース作成ウィザードで生成されたロード・プログラムを選択し、最初の作成のデータをロードします。
選択するプログラムが不明の場合、「**Dimension**」フォルダを開いて「**cubedef**」ディメンションを選択します。「**Properties**」ページで、AW\$LOADPRGS プロパティの値を記録します。
3. プロパティ・ビューアの「**Program**」ページで、ロード・プログラムを編集し、ソース表の名前を変更します。
4. 「**Apply**」、「**Compile**」の順に選択します。
続行する前にエラーをすべて修正します。

- ロード・プログラムの名前を右クリックし、「Copy to Clipboard」を選択します。
- OLAP Worksheet を開き、プログラムを実行します。[Ctrl]+[V] を押して、問合せウィンドウにプログラム名を貼り付けます。

```
CALL program
```

- LIMIT および REPORT コマンドを使用して新しいデータをチェックします。
- UPDATE および COMMIT コマンドを発行して新しいデータを保存します。

事歴 : Units キューブのリフレッシュ

Global スター・スキーマでは、別々の更新表に月のデータを追加します。

- オブジェクト・ビューで、「Programs」フォルダを開き、「GLOBAL_AW.GLOBAL!___GET.CUBE.DATA_UNITS_CUBE_1」を選択します。
- ロード・プログラムの「Program」ページを表示し、ソース・ファイルの名前「UNITS_HISTORY_FACT」を見つけます。

```
SQL DECLARE C1 CURSOR FOR SELECT CHANNEL_ID,SHIP_TO_ID,ITEM_ID, -
MONTH_ID,UNITS FROM GLOBAL.UNITS_HISTORY_FACT
```

- この文を編集して UNITS_HISTORY_FACT を UNITS_UPDATE_FACT に置き換えます。
- 「Apply」、「Compile」の順に選択します。
- OLAP Worksheet を開き、次のようなコマンドで修正したロード・プログラムを実行します。

```
CALL ___get.cube.data_units_cube_1
```

- 次のようなコマンドで新しいデータのサンプルを表示します。

```
LIMIT channel TO '1'          "Select any channel
LIMIT product TO '1'         "Select any product
LIMIT time TO '91'           "Select the new time period
LIMIT customer TO '76'       "Select any customer
" Add the parent values of customer 76
LIMIT customer ADD ANCESTORS USING customer_parentrel
REPORT DOWN customer units  "View the new data
```

```
CHANNEL: 1
PRODUCT: 1

--UNITS---
---TIME---
CUSTOMER      91
-----
76              1,220.00
```

17	7,378.00
8	12,985.00
1	50,632.00

7. 次のコマンドで変更を保存します。

```
UPDATE  
COMMIT
```

または

Analytic Workspace Manager の「File」メニューから「**Save**」を選択します。

データのリフレッシュで再有効化が必要な場合

データのルーチン・リフレッシュでは、特定のアプリケーションに対してワークスペースを再有効化する必要はありません。これは、イネーブラによって作成されたビューは、新しいディメンション・メンバーに対して再定義する必要がないからです。ただし、ワークスペースを再有効化する必要がないのは、論理モデルに次のような変更を加えた場合です。

- ソース・キューブの OLAP カタログ・メタデータの変更
- アナリティック・ワークスペース内のキューブの追加または削除
- アナリティック・ワークスペース内のメジャーの追加または削除
- アナリティック・ワークスペース内の階層の追加または削除
- アナリティック・ワークスペース内のレベルの追加または削除
- アナリティック・ワークスペースの OLAP カタログ・メタデータの変更

有効化を行う手順はあまり時間がかからないので、リフレッシュするたびにアナリティック・ワークスペースを再有効化の方がいいかもしれません。

古いディメンション・メンバーを削除するか（新たに加えた期間と同じ数の古い期間を排除するなど）、新しいデータ値のみロードする場合、作成スクリプトを生成して DBMS_AWM パッケージへのコールを変更します。

参照： DBMS_AWM パッケージの詳細は、『Oracle OLAP リファレンス』を参照してください。

アナリティック・ワークスペースへの SQL アクセス

この章では、SQL を使用してアナリティック・ワークスペースのデータにアクセスする方法について説明します。これらの方法のほとんどは、アプリケーションの一部として実行時に使用できます。

この章では、次の項目について説明します。

- [SQL アクセスの概要](#)
- [DBMS_AW_UTILITIES を使用したカスタム・メジャーの作成](#)
- [事例: DBMS_AW_UTILITIES を使用した、Global への Sales の追加](#)
- [OLAP_EXPRESSION を使用したカスタム・メジャーの作成](#)
- [事例: OLAP_EXPRESSION を使用した、Global への Sales の追加](#)
- [OLAP_TABLE を使用した、ワークスペース・データへの直接アクセス](#)
- [事例: OLAP_TABLE を使用した Global カスタム・メジャーの作成](#)

SQL アクセスの概要

SQL を使用すると、アナリティック・ワークスペースのデータを操作したり、そのデータをアプリケーションに抽出できます。使用できる方法は様々で、最良の方法はアナリティック・ワークスペースのタイプ、目的とするタスクおよび個人の環境によって異なります。

アナリティック・ワークスペースのデータの操作

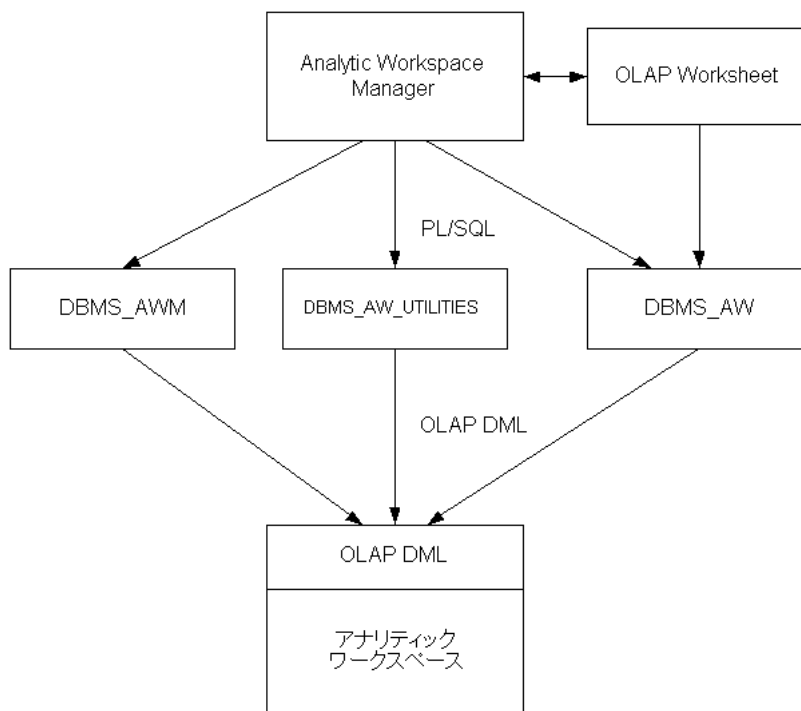
SQL を使用してアナリティック・ワークスペースのデータを操作するには、OLAP DML コマンドを実行する PL/SQL プロシージャを使用する必要があります。OLAP DML は、アナリティック・ワークスペースを操作するための言語です。これを使用すると、ワークスペース・オブジェクトを作成、変更、削除および移入できます。これらのタスクを実行するいずれの方法でも OLAP DML を使用します。

OLAP DML コマンドを実行する PL/SQL パッケージは複数あります。1つのプロシージャへのコールによって、1つまたは多数の OLAP DML コマンドを実行でき、特定のタスクを実行します。このようなパッケージには、次のものがあります。

- DBMS_AW には、個々の OLAP DML コマンドを実行するプロシージャが含まれています。
- DBMS_AW_UTILITIES には、OLAP API および BI Beans に対して有効化されているスタンダード・フォームのアナリティック・ワークスペースのカスタム・メジャーを管理するプロシージャが含まれています。
- DBMS_AWM には、スタンダード・フォームのアナリティック・ワークスペースを作成するプロシージャが含まれています。

これらのパッケージは、SQL*Plus などの SQL インタフェースで直接使用できます。Analytic Workspace Manager および OLAP Worksheet は、これらの SQL パッケージを使用するアプリケーションです。図 7-1 に、これらの関係を示します。

図 7-1 Analytic Workspace Manager での PL/SQL パッケージの使用



アナリティック・ワークスペースの問合せ

OLAP_TABLE ファクションは、アナリティック・ワークスペースを問い合わせるための基本的なテクノロジーを提供します (7-11 ページの「[OLAP_TABLE を使用した、ワークスペース・データへの直接アクセス](#)」を参照)。これは、スタンダード・フォームの規則の外部で機能し、すべてのアナリティック・ワークスペースのデータにアクセスできます。ただし、イネーブラなど、OLAP_TABLE を使用するツールで適切な構文を構成するには、スタンダード・フォームが必要です。

アクティブ・カタログについて

Oracle OLAP は、スタンダード・フォームのアナリティック・ワークスペースに関する情報のカタログを提供します。この**アクティブ・カタログ**は、自動的に生成およびメンテナンスされ、DBA によるアクションは不要です。

アクティブ・カタログは、名前が ALL_OLAP2_AW で始まるパブリック・ビューとして実装されます。たとえば、ALL_OLAP2_AW_CUBES はすべてのアナリティック・ワークスペースのキューブを表示し、ALL_OLAP2_AW_DIMENSIONS はすべてのディメンションを表示します。アクティブ・カタログは、SQL から直接問い合わせることができます。

アクティブ・カタログの詳細は、『Oracle OLAP リファレンス』を参照してください。

カスタム・メジャーのサポート

カスタム・メジャーは、アナリティック・ワークスペースに格納されている 1 つ以上のメジャーから計算されます。多くの場合、セッションの期間に限定して分析者が作成します。ただし、カスタム・メジャーは、アナリティック・ワークスペースの永続的な部分として保存することもできます。

こうして保存したカスタム・メジャーは、実行時に解決したり、変数に格納できます。実行時の計算では、ディスクの記憶域が必要になることも、データ・メンテナンスに必要な処理時間が長くなることもありません。ただし、パフォーマンスは低下する可能性があります。そこで、どのメジャーをオンデマンドで計算し、どのメジャーを変数に格納するか（格納する必要がある場合）を決定する必要があります。この章で説明するカスタム・メジャーは問合せのために計算されます。ストアド・カスタム・メジャーの作成手順については、[第 9 章](#)を参照してください。

カスタム・メジャーの定義方法

アナリティック・ワークスペースのカスタム・メジャーは、次の 2 つの PL/SQL パッケージでサポートされています。

- DBMS_AW_UTILITIES には、カスタム・メジャーを作成、更新および削除するプロシージャが含まれています。このパッケージは、OLAP API および BI Beans のイネーブラによって作成されたビューでのみ機能します。カスタム・メジャーは、カスタム・メ

ジャーのこれらのビューで提供される事前定義済の列に格納されます。カスタム・メジャーは、セッションの期間中のみ有効か、永続的に有効なものを定義できます。

- DBMS_OLAP では、OLAP DML コマンドを実行する様々なプロシージャが含まれています。その一部は、SELECT 文で使用し、アナリティック・ワークスペースにおける計算またはデータ操作を実行できます。計算結果は、他の結果セットとともに戻されます。このタイプのカスタム・メジャーは、SELECT 文の有効期間にのみ存在します。

また、OLAP_TABLE ファンクションを使用すると、スタンダード・フォームのフレームワークの外部でカスタム・メジャーを定義したり、カスタム・メジャーにアクセスできます (7-11 ページの「[OLAP_TABLE を使用した、ワークスペース・データへの直接アクセス](#)」を参照)。

カスタム・メジャーの分析サポート

カスタム・メジャーの定義方法にかかわらず、計算自体は OLAP DML を使用して表現します。ここでは、データ操作に使用できる多くのファンクションおよびコマンドについて説明します。また、乗算 (*)、除算 (/)、加算 (+)、減算 (-) などの演算子を使用して行をまたぐ計算を実行できます。

予測および回帰

OLAP DML では、最も優れた最新の予測および回帰のツールが提供されています。このツールには、単純な線形回帰、非線形回帰法、単一指数平滑法、二重指数平滑法および Holt-Winters 法が含まれます。

時系列操作

時系列ファンクションは、リード、ラグ、移動平均などの操作を実行します。[表 7-1](#) に、簡単にカスタム・メジャーに組み込むことができる時系列ファンクションを示します。

表 7-1 OLAP DML の時系列ファンクション

ファンクション	戻される内容
CUMSUM	累計
LAG	指定したオフセットでの前の期間の値
LAGABSPCT	ある値と、指定したオフセットでの前の期間の絶対値とのパーセントによる差
LAGDIF	ある値と、指定したオフセットでの前の期間の値との差
LAGPCT	ある値と、指定したオフセットでの前の期間の値とのパーセントによる差
LEAD	指定したオフセットでの以降の期間の値
MOVINGAVERAGE	指定した範囲での一連の平均値

表 7-1 OLAP DML の時系列ファンクション

ファンクション	戻される内容
MOVINGMAX	指定した範囲での一連の最大値
MOVINGMIN	指定した範囲での一連の最小値
MOVINGTOTAL	指定した範囲での一連の合計

財務演算

財務ファンクションには、スプレッドシートで指定するものと同様に、金利の計算、減価償却および支払い明細が含まれます。

たとえば、FPMTSCHED ファンクションでは、指定された期間に定率のローンを返済するために、支払い明細（元金および利息）を計算します。次の FPMTSCHED へのコールでは、LOANS 変数にリストされた金額に基づいて、RATES 変数にリストされた金利で、これらの変数の MONTH ディメンションに対して、36 回払いを計算します。

```
FPMTSCHED(loans, rates, 36, month)
```

統計演算

統計演算には、標準偏差、ランクおよび相関が含まれます。たとえば、STDDEV ファンクションは標準偏差を計算します。たとえば、次のファンクション・コールがあります。

```
STDDEV(units month)
```

このファンクション・コールは、現在選択されているすべての月について、UNITS メジャーの値の標準偏差を戻します。

数値計算

ファンクションを使用すると、様々な計算（サイン、コサイン、平方根、最小、最大）およびデータ型変換を実行できます。

たとえば、MAX ファンクションは 2 つの式を比較して、大きい方の値を戻します。たとえば、次のファンクション・コールがあります。

```
MAX(actual, forecast)
```

このファンクション・コールは、ACTUAL メジャーと FORECAST メジャーを比較して、現在選択されているすべてのディメンション・メンバーについて大きい方の値を戻します。

テキスト操作

OLAP DML では、文字列の連結、サイズが大きいテキスト本文への文字列の配置、文字列の挿入などを実行するファンクションを使用して、シングलバイトおよびマルチバイトのキャラクタ・セットを操作できます。

たとえば、EXTCHARS ファンクションはテキストの一部を抽出します。たとえば、次のファンクション・コールがあります。

```
EXTCHARS('lastname,firstname', 1,8)
```

このファンクション・コールは最初の 8 文字を抽出します。抽出される文字列は次の 8 文字です。

```
lastname
```

アロケーション

アロケーションは、プランニング・アプリケーションの重要な部分です。マネージャは、販売割当て、製品の成長、給与、設備など、組織の目標を設定し、その目標をそれぞれの関係者に割り当てる必要があります。サポートされているアロケーション・メソッドには、次のものが含まれます。

- コピー・メソッド（階層のコピー、最小、最大、最初、最後）
- 均等配賦（均等、階層均等）
- 比例配賦（加重配賦、ユーザー定義の多次元ファンクション）

集計

集計は、アナリティック・ワークスペースの基本的な機能です。スタンダード・フォームのアナリティック・ワークスペースを作成すると、各キューブに対するデフォルトの集計プランが含まれます。Analytic Workspace Manager のウィザードを使用すると、格納済の集計レベルをすばやく簡単に識別できます。

OLAP DML は、Analytic Workspace Manager または PL/SQL プロシージャで現在使用できる集計メソッドより守備範囲の広い集計メソッドを提供します。レベル、個々のメンバー、メンバー属性、時間範囲、データ値などの基準に基づいて、適切なメソッドを選択できます。

モデル

モデルは、相互に関連のある一連の方程式です。次に、OLAP DML でサポートされるモデリング機能の一部を示します。

- 一意の計算規則に従って、個々のディメンション・メンバーに対して計算を実行できます。
- 計算順序は Oracle OLAP が判断するため、依存関係を考えずに任意の順に指定できます。
- Oracle OLAP が連立方程式を解きます。

結果は、変数またはディメンション・メンバーのいずれかに割り当てることができます。ディメンションベースの方程式は柔軟性があります。モデルを解決するまでモデリング変数

を指定する必要がないため、同じディメンションを持つ他のメジャーで、同じモデルを実行できます。たとえば、いずれも Line ディメンションを持つ Budget および Actual で、同じモデルを実行できます。

DBMS_AW_UTILITIES を使用したカスタム・メジャーの作成

OLAP API および BI Beans のイネーブラでは、DBMS_AW_UTILITIES パッケージで定義したカスタム・メジャー専用の列を持つファクト・ビューを作成します。数値データの列が 100 列あり、CUST_MEAS_NUM1 から CUST_MEAS_NUM100 という名前が付けられています。また、テキスト・データの列が 100 列あり、CUST_MEAS_TEXT1 から CUST_MEAS_TEXT100 という名前が付けられています。

次に、カスタム・メジャーを作成する場合の基本構文を示します。

```
CALL DBMS_AW_UTILITIES.CREATE_CUSTOM_MEASURE(  
    schema.aw_name, aw_formula_name, aw_formula_expression,  
    'PERMANENT'|'TEMPORARY', schema.view_name;
```

OLAP API イネーブラでは、アナリティック・ワークスペースのビューの CWM2 メタデータを作成し、DBMS_AW_UTILITIES では、これらのビューに追加されたカスタム・メジャーの CWM2 メタデータを作成します。このメタデータは、ビューの一般的な列の名前とカスタム・メジャーとの間のマッピングを識別する表に格納されます。

- CWM2\$_AW_TEMP_CUST_MEAS_MAP は、現行のユーザーの一時的なカスタム・メジャーを示します。
- CWM2\$_AW_PERM_CUST_MEAS_MAP は、DBA ロールを持つユーザーの永続的なカスタム・メジャーを示します。

事例 : DBMS_AW_UTILITIES を使用した、Global への Sales の追加

3-5 ページの「必要とされるビジネス・ファクトの識別」では、Global Corporation のデータ要件を識別しています。スター・スキーマに格納されるファクトは 3 つのみで、それ以外はアナリティック・ワークスペースで計算する必要があります。GLOBAL は OLAP API に対して有効化されているスタンダード・フォームのアナリティック・ワークスペースであるため、DBA は DBMS_AW_UTILITIES パッケージを使用して、これらのメジャーを定義できます。

アナリティック・ワークスペースに関する情報の取得

カスタム・メジャーを定義するには、アナリティック・ワークスペースですでに定義されているメジャーの名前を知っている必要があります。GLOBAL アナリティック・ワークスペースに定義されているメジャーの名前については、アクティブ・カタログ内の ALL_OLAP2_AW_CUBE_MEASURES ビューを問い合わせます。例 7-1 に、メジャーの名前を取得する方法を示します。

例 7-1 メジャー名に関するアクティブ・カタログの問合せ

```
SELECT aw_cube_name, aw_measure_name
FROM all_olap2_aw_cube_measures
WHERE aw_owner = 'GLOBAL_AW' AND
aw_name = 'GLOBAL';
```

AW_CUBE_NAME	AW_MEASURE_NAME
PRICE_CUBE	UNIT_COST
PRICE_CUBE	UNIT_PRICE
UNITS_CUBE	UNITS

ALL_AW_CUBE_ENABLED_VIEWS ビューでは、OLAP API に対して有効化されているキューブ、これらのキューブにアクセスするためにイネーブラで作成されたビューの名前、各ビューのディメンションおよびディメンション階層を識別します。

例 7-2 では、Price キューブが PRODUCT および TIME でディメンション化され、GLOB_GLOBA_PRICE_CU4VIEW というビューで問合せできることを示します。Units キューブは、CHANNEL、CUSTOMER、PRODUCT および TIME でディメンション化されています。CUSTOMER ディメンションは、MARKET_SEGMENT (GLOB_GLOBA_UNITS_CU9VIEW 内) および SHIPMENTS (GLOB_GLOBA_UNITS_CU10VIEW 内) の 2 つの階層を持ちます。

例 7-2 アクティブ・カタログを問い合わせる SELECT 文

```
SELECT cube_name, system_viewname, hiercombo_str
FROM all_aw_cube_enabled_views WHERE
aw_name = 'GLOBAL' AND
cube_name = 'PRICE_CUBE' OR
cube_name = 'UNITS_CUBE';
```

CUBE_NAME	SYSTEM_VIEWNAME	HIERCOMBO_STR
PRICE_CUBE	GLOB_GLOBA_PRICE_CU4VIEW	DIM:PRODUCT/HIER:PRODUCT_ROLLUP;DIM:TIME/HIER:Calendar
UNITS_CUBE	GLOB_GLOBA_UNITS_CU9VIEW	DIM:CHANNEL/HIER:CHANNEL_ROLLUP; DIM:CUSTOMER/HIER:MARKET_SEGMENT; DIM:PRODUCT/HIER:PRODUCT_ROLLUP; DIM:TIME/HIER:Calendar
UNITS_CUBE	GLOB_GLOBA_UNITS_CU10VIEW	DIM:CHANNEL/HIER:CHANNEL_ROLLUP; DIM:CUSTOMER/HIER:SHIPMENTS; DIM:PRODUCT/HIER:PRODUCT_ROLLUP; DIM:TIME/HIER:Calendar

DBMS_AW_UTILITIES を使用した、Sales のカスタム・メジャーとしての定義

カスタム・メジャーの定義に必要な情報を取得した後、DBMS_AW_UTILITIES を使用してカスタム・メジャーを定義できます。この例では、他の 2 つのメジャーの積を計算する

SALES、ディメンション・メンバーの各組合せに対して UNITS および UNIT_PRICE を定義します。

UNITS は Units キューブのメジャーで、UNIT_PRICE は Price キューブのメジャーです。Units キューブは、TIME、PRODUCT、CUSTOMER および CHANNEL の 4 つのディメンションを持ちます。Price キューブが持つディメンションは、TIME および PRODUCT の 2 つのみです。これら 2 つのメジャーの積は 4 つのディメンションを持つため、SALES は Units キューブのビューに追加する必要があります。

例 7-3 では、Units キューブの両方のビューに SALES メジャーを追加します。SALES 式の方程式を指定しているのは、最初のコールのみであることに注意してください。2 番目のコールは単に、既存の SALES 式を識別します。

注意： SQL*Plus などの SQL 環境で DBMS_AW_UTILITIES を使用する場合、必ず次の設定で開始します。

```
SET SERVEROUT ON
EXECUTE CWM2_OLAP_MANAGER.SET_ECHO_ON
```

この設定で開始しないと、診断メッセージが表示されません。

例 7-3 DBMS_AW_UTILITIES を使用した SALES の定義

```
SET SERVEROUT ON
EXECUTE CWM2_OLAP_MANAGER.SET_ECHO_ON
EXECUTE DBMS_AW_UTILITIES.CREATE_CUSTOM_MEASURE(
    'global_aw.global', 'sales', 'units * unit_price',
    'PERMANENT', 'global_aw.glob_globa_units_cu9view');

EXECUTE DBMS_AW_UTILITIES.CREATE_CUSTOM_MEASURE(
    'global_aw.global', 'sales', '',
    'PERMANENT', 'global_aw.glob_globa_units_cu10view');
```

ワークスペースの式の表示

アナリティック・ワークスペースで作成した式を表示するには、次のコマンドを使用します。

```
EXECUTE DBMS_AW.EXECUTE('DESCRIBE sales');

DEFINE SALES FORMULA DECIMAL <TIME CUSTOMER PRODUCT CHANNEL>
EQ units * unit_price
```

また、Analytic Workspace Manager で SALES のプロパティ・シートを表示することもできます。

Sales カスタム・メジャーの問合せ

OLAPSYS.CWM2\$_AW_PERM_CUST_MEAS_MAP は、SALES カスタム・メジャーとビューの列間のマッピングを識別します。

```
SELECT aw_access_view_name, cust_adt_column, aw_measure_name
FROM olapsys.cwm2$_aw_perm_cust_meas_map
WHERE workspace_name = 'global_aw.global';
```

AW_ACCESS_VIEW_NAME	CUST_ADT_COLUMN	AW_MEASURE_NAME
global_aw.glob_globa_units_cu9view	CUST_MEAS_NUM1	sales
global_aw.glob_globa_units_cu10view	CUST_MEAS_NUM1	sales

SALES メジャーの問合せでは、2 つの表の CUST_MEAS_NUM1 列から値を選択する必要があります。

OLAP_EXPRESSION を使用したカスタム・メジャーの作成

DBMS_AW パッケージには、数値の実行時計算を指定する OLAP_EXPRESSION プロシージャが含まれています。

このプロシージャでは、アナリティック・ワークスペース・データのビューに対して SELECT 文を使用して行間計算を指定できます。計算は、OLAP エンジンによって実行されます。OLAP_EXPRESSION を使用する唯一の要件は、ビューの SELECT 文に、ROW2CELL 句を持つ OLAP_TABLE ファンクションへのコールを含めることです。このタイプのビューは、OLAP API および BI Beans のイネーブラで生成し、ROW2CELL 列を持つカスタム・ビューを生成することもできます (7-11 ページの「[OLAP_TABLE を使用した、ワークスペース・データへの直接アクセス](#)」を参照)。

次に、OLAP_EXPRESSION の基本構文を示します。

```
OLAP_EXPRESSION(r2c, expression)
```

例: OLAP_EXPRESSION('R2C', 'units * unit_price')

事例 : OLAP_EXPRESSION を使用した、Global への Sales の追加

OLAP API に対する有効化によって、CUSTOMER ディメンションの 2 つの各階層に 1 つずつ Units キューブのビューが 2 つ作成されています。次の SELECT 文は、いずれかのビューを問い合わせ、Sales に新しい列を生成します。SALES 列は、アナリティック・ワークスペースで計算されます。

```
SELECT time_et, units, OLAP_EXPRESSION(r2c, 'units * unit_price') sales
FROM global_aw.glob_globa_units_cu9view WHERE
channel_et = '1' AND
product_et = '4' AND
```



```
customer_et = '24' AND
time_et > '66' AND
units IS NOT NULL
ORDER BY OLAP_EXPRESSION(r2c, 'units * unit_price') DESC;
```

この SELECT 文の結果セットは、Sales が降順で示されるようソートされます。

TIME_ET	UNITS	SALES
8	6	170017.38
68	5	123300.25
9	3	93293.85
7	3	64931.7
67	2	50932.26

OLAP_TABLE を使用した、ワークスペース・データへの直接アクセス

OLAP_TABLE ファンクションは、アナリティック・ワークスペースからデータを抽出するための基本的なテクノロジーを提供します。イネーブラで生成されるアナリティック・ワークスペースのビューはすべて、OLAP_TABLE ファンクションを使用します。OLAP_TABLE を直接使用することにより、データ・アクセスを完全に制御できます。また、イネーブラがないアプリケーションをサポートする独自のビューを開発して、ワークスペース・データをアプリケーションに直接抽出できます。この機能によって、ユーザーは、実行時における OLAP_TABLE へのコールの中に問合せを作成できるので、アプリケーションの柔軟性が飛躍的に向上します。

OLAP_TABLE ファンクションを使用する OLAP ツールではスタンダード・フォームのアナリティック・ワークスペースが必要ですが、OLAP_TABLE ファンクション自体はスタンダード・フォームを必要としません。

Oracle のオブジェクト・リレーショナル・モデルについて

Oracle のオブジェクト・リレーショナル・モデルでは、SQL から多次元データにアクセスするための基礎となるテクノロジーを提供します。オブジェクト・リレーショナル・モデルには、オブジェクト型およびテーブル・ファンクションというこのアクセスを実現するための 2 つの重要なメカニズムが含まれます。

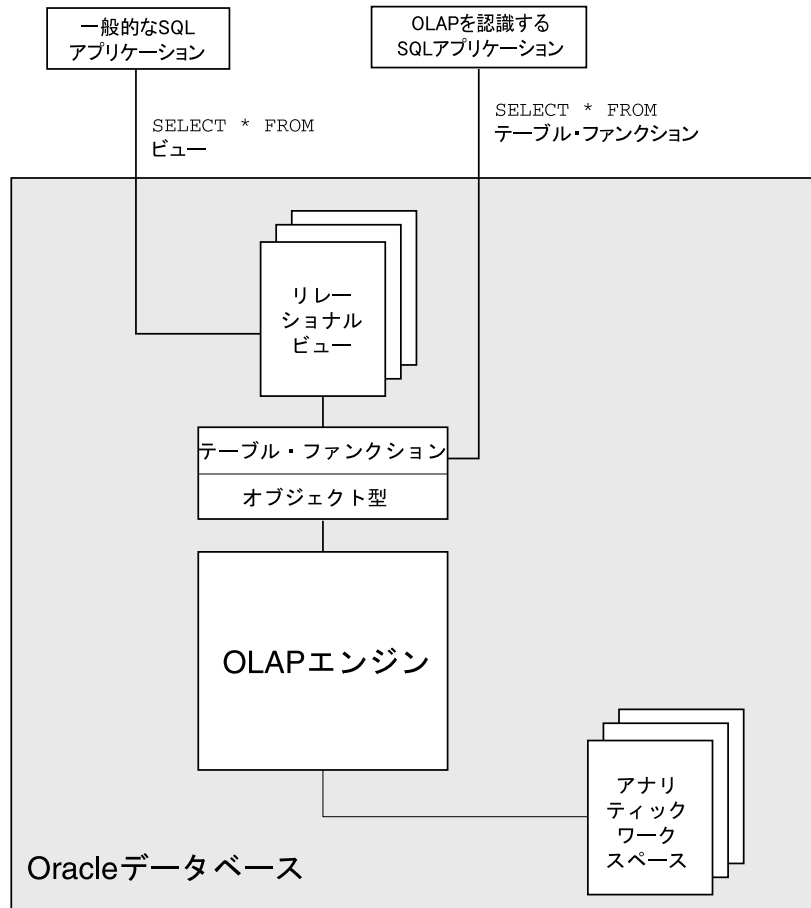
- **オブジェクト型** (抽象データ型または ADT と呼ぶ) は、PL/SQL におけるオブジェクト指向プログラミングの基礎となるものです。オブジェクト型は、データ構造と、データの操作に必要なファンクションおよびプロシージャをカプセル化します。CREATE TYPE 文を使用してオブジェクト型を定義する場合は、実在するエンティティに関連付けられた抽象テンプレートを作成します。

アナリティック・ワークスペースでは、実在するエンティティはディメンション、変数および式です。したがって、これらのデータ・コンテナのオブジェクト型を定義します。これらのオブジェクト型の定義では、多次元データの形式を行および列として SQL に定義します。

- **テーブル・ファンクション**は、入力として行セットを取り、物理的なデータベース表と同様に問い合わせることができる出力として行セットを作成します。問合せの FROM 句では、データベース表の名前ではなく、テーブル・ファンクションを使用します。

OLAP_TABLE ファンクションは、ワークスペース・データが格納されている LOB からデータを抽出し、表を作成する目的で開発されました。OLAP_TABLE は、データ・オブジェクトを選択し、操作してビューの列として戻す OLAP エンジンに渡されるパラメータを受け入れます (図 7-2 を参照)。

図 7-2 多次元データへの SQL アクセス



参照： PL/SQL テーブル・ファンクションについては、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

アナリティック・ワークスペースのビューの設計

作成するビューの数、ビュー内の列の数および特性は、これらのビューがサポートするよう設計されているアプリケーションの要件によって大きく異なります。

アナリティック・ワークスペースには集計データが含まれるため、ビューに集計を含める必要があります。集計データを表示する形式を次に示します。

- ディメンション・ビューおよびメジャー・ビューを使用してスター・スキーマを作成します。ディメンション・ビューは、単一列のすべてのレベルのディメンション・メンバーを示します。
- すべてのディメンション、属性およびメジャーの列を含むビューを作成します。
- ロールアップ形式のビューを作成します。この形式では、複数列の各ディメンション・メンバーの完全な親子関係が示されます。
- 各集計レベルで個別の表を使用します。

これらのビュー・タイプの例については、『Oracle OLAP リファレンス』を参照してください。

プロセスの概要

次に、アナリティック・ワークスペースに格納されているデータのビューを作成するために実行する必要がある基本手順を示します。

1. アナリティック・ワークスペースを調査し、アプリケーションに表示する変数、式、リレーションおよびディメンションを識別します。
2. アプリケーションの要件に基づいて、リレーショナル表またはビュー内のこれらのオブジェクトの表示方法を決定します。
3. 作成する表またはビューごとに、次の手順を実行します。
 - a. 表内の行についてオブジェクト型を作成します。このオブジェクト型で表の列を定義します (7-14 ページの「[オブジェクト型の作成](#)」を参照)。
 - b. このオブジェクト型で構成される表を作成します (7-15 ページの「[オブジェクトの表の作成](#)」を参照)。
 - c. OLAP_TABLE ファンクションを使用して SELECT 文を発行し、表を移入します。SELECT 文は、CREATE VIEW 文への引数にすることができます (7-15 ページの「[オブジェクトの表の作成](#)」および 7-17 ページの「[アナリティック・ワークスペース・データのビューの作成](#)」を参照)。
4. データベースへの変更をコミットします。汎用のビューを作成する場合、この手順は必須です。データを直接アプリケーションに取得する場合、この手順はオプションです。ただし、将来使用するためにオブジェクト型および表型の定義を保存することもできます。
5. ビューの場合、ビューを問い合わせるためにアプリケーションに必要なメタデータをすべて作成します。

オブジェクト型の作成

オブジェクト型は、属性（表の列と同等）で構成されます。最終的にリレーショナル・ビューを作成する場合は、これらの属性からそのビューの列を選択します。ただし、オブ

ジェクト型および属性というより、行および列という方が、このプロセスを理解しやすくなります。オブジェクト型を作成することで表の行を定義します。

次に、行を定義する場合の基本構文を示します。

```
CREATE TYPE row AS OBJECT (  
    column1      datatype,  
    column2      datatype,  
    columnn      datatype,  
    row2cell     RAW(32));
```

オブジェクトの表の作成

表型はオブジェクトの集合です。次に、表を作成する場合の基本構文を示します。

```
CREATE TYPE table_name AS TABLE OF object_name;
```

OLAP_TABLE ファンクションの構文

次に、OLAP_TABLE ファンクションの基本構文を示します。

```
(OLAP_TABLE(  
    'aw_name DURATION time'  
    'table_name'  
    'olap_command'  
    'limit_map');
```

構文で示されているように、OLAP_TABLE ファンクションは4つのテキスト・パラメータを取ります。これらのパラメータはOLAP エンジンに渡されて次の処理で使用されます。

- アナリティック・ワークスペースへの接続手順
- データのターゲットとなるオブジェクトの表の名前
- OLAP DML コマンド (オプション)
- 制限マップ

参照： OLAP_TABLE ファンクションの構文の詳細は、『Oracle OLAP リファレンス』を参照してください。

接続手順

OLAP_TABLE ファンクションの最初のパラメータは、ソース・データが格納されるアナリティック・ワークスペースの名前を提供し、OLAP セッションにアナリティック・ワークスペースがアタッチされている時間を指定します。アナリティック・ワークスペースは、問合せの終了時またはセッションの終了時にデタッチできます。次に、このパラメータの完全な構文を示します。

```
'aw_name DURATION QUERY | SESSION'
```

次に例を示します。

```
'global DURATION QUERY'
```

SESSION を指定する場合は、アナリティック・ワークスペースがすでにアタッチされているため、OLAP_TABLE への後続のコールでこのパラメータに空の文字列を使用できます。ただし、接続文字列は必要以上に繰り返すと無視されます。

SESSION を指定すると、アナリティック・ワークスペースがセッションに 1 回のみアタッチされるため、QUERY より高いパフォーマンスが提供されます。

表型

2 番目のパラメータは、定義したオブジェクトの表名を識別します (7-15 ページの「[オブジェクトの表の作成](#)」を参照)。次に、このパラメータの完全な構文を示します。

```
'table_name'
```

次に例を示します。

```
'sales_table'
```

OLAP DML コマンド

OLAP_TABLE ファンクションの 3 番目のパラメータは、単一の OLAP DML コマンドです。複数のコマンドを実行する場合は、アナリティック・ワークスペースにプログラムを作成し、このパラメータでそのプログラムを指定します。

通常、このパラメータは、1 つ以上のディメンションを制限する場合に使用します。ディメンション・メンバーの選択は、OLAP_TABLE へのコールの実行中にのみ有効です。したがって、それ以外の OLAP セッションは影響を受けません。

次に、このパラメータの完全な構文を示します。

```
'olap_command'
```

次に例を示します。

```
'LIMIT customer_hierlist TO ''SHIPMENTS'''
```

このパラメータの機能および柔軟性は、OLAP DML で使用可能なデータ操作コマンドを仮想的に処理する機能に基づいています。

制限マップ

OLAP_TABLE ファンクションの 4 番目（最後）のパラメータは、アナリティック・ワークスペース・オブジェクトを表の列にマップし、各オブジェクトのロールを識別します。このパラメータは、SQL の SELECT 文の WHERE 句と組み合されて、一連の LIMIT コマンドをアナ

リティック・ワークスペースに生成するため、制限マップと呼ばれます。制限マップの内容はビューを定義します。

制限マップの構文には、主にディメンション階層を定義するために、多くの句が含まれています。これらの句の多くは、条件に応じてオプションになります。したがって、例を示しながら構文を説明の方が効果的です。ただし、この章では使用可能なすべての句を説明しているわけではありません。

制限マップは、アナリティック・ワークスペースに永続的に格納できます。制限マップは、OLAP API および BI Beans のイネーブラによって、OLAP_SYS_LIMITMAP というテキスト変数に格納されます。このような制限マップおよびそれを使用したビュー定義を調べると参考になります。

また、Analytic Workspace Manager で有効化スクリプトを作成して、OLAP_TABLE ファンクションの使用方法を学習できます。

アナリティック・ワークスペース・データのビューの作成

アナリティック・ワークスペースのビューは、保存された SELECT 文であるという点では他のリレーショナル・ビューと同様です。SELECT 構文でリレーショナル表の名前のかわりに、OLAP_TABLE ファンクションが使用される点が違います。

次に、OLAP_TABLE ファンクションを使用してビューを作成する場合の基本構文を示します。CAST ファンクションは、OLAP_TABLE で頻繁に使用しますが、必須ではありません。

```
CREATE OR REPLACE VIEW view_name AS
SELECT columns
FROM TABLE(CAST (OLAP_TABLE(
    'aw_name DURATION time'
    'table_name'
    'olap_command'
    'limit_map'))
AS table_name);
```

ビュー用のスクリプトの作成

ビューを作成するには、テキスト・エディタを使用して、行、表およびビューを定義する PL/SQL スクリプトを作成します。例 7-4 は、アナリティック・ワークスペースのビュー用に SQL スクリプトを作成するためのテンプレートです。このスクリプトは、SQL*Plus の @ コマンドを使用して実行できます。

例 7-4 ビュー作成用テンプレート

```
SET ECHO ON
SET SERVEROUT ON

DROP TYPE table;
```

```
DROP TYPE row;

CREATE TYPE row AS OBJECT (
    column1      datatype,
    column2      datatype,
    columnn      datatype);
/
CREATE TYPE table AS TABLE OF row;
/
CREATE OR REPLACE VIEW view AS
SELECT column1, column2, columnn
FROM TABLE(CAST(OLAP_TABLE(
    'connection',
    'table',
    'olap_command',
    'limit_map')
AS table));
/
COMMIT
/
GRANT SELECT ON view TO PUBLIC;
```

事例 : OLAP_TABLE を使用した Global カスタム・メジャーの作成

Global Corporation では、スター・スキーマから GLOBAL アナリティック・ワークスペースにフェッチした 3 つのストアド・メジャーの他にも、多くのカスタム・メジャーを必要とします。OLAP_TABLE ファンクションは、これらの導出メジャーを作成する方法を提供します。ただし、GLOBAL では、この章で前述した他の方法も使用できます。

UNITS はストアド・メジャーの 1 つです。前の期間の UNITS は必須の導出メジャーです。前の期間との差または変化率などの導出メジャーは、必須ではありませんが、作成することが可能です。

導出メジャーは、アナリティック・ワークスペースで永続的に定義したり、OLAP_TABLE ファンクションの構文で指定できます。例では、次の 2 つのメジャーを追加します。

- UNITS_PP は、前の期間の販売単位を計算します。
- UNITS_PCTCHG_PP は、前の期間からの変化率です。

例では、これらのメジャーの OLAP カタログ・キューブを新規作成します。

アナリティック・ワークスペースにおける式の定義

次のコマンドで UNITS_PP をアナリティック・ワークスペースに追加します。


```
DEFINE units_pp FORMULA LAG(units, 1, time, LEVELREL time_levelrel)
UPDATE;COMMIT
```

式を定義する構文では、ソース・オブジェクトと同じデータ型およびディメンションを指定します。次に、新しい式の定義を示します。

```
DEFINE UNITS_PP FORMULA DECIMAL <TIME CUSTOMER PRODUCT CHANNEL>
EQ lag(units, 1, time, levelrel time_levelrel)
```

また、Analytic Workspace Manager のプロパティ・シートを使用して、UNITS_PP を定義することもできます。

例 7-7 では、OLAP DML LAGPCT ファンクションを使用して、OLAP_TABLE ファンクションに UNITS_PCTCHG_PP を定義します。

UNITS_PP と UNITS_PCTCHG_PP は、いずれもスタンダード・フォームのメジャーとしては定義されません。スタンダード・フォームに準拠するには、複数の OLAP DML プロパティが割り当てられ、スタンダード・フォームのカatalogにメジャーとして登録される必要があります。ただし、OLAP_TABLE および OLAP Catalog はスタンダード・フォームを必要としません。これらの使用を簡素化するツールでスタンダード・フォームが必要になります。

OLAP_TABLE を使用したアナリティック・ワークスペースの問合せ

例 7-5 に、OLAP_TABLE ファンクションを持つ SELECT 文を使用して、データを直接 SQL アプリケーションにフェッチするスクリプトを示します。この選択は、アプリケーション有効化プロセスとは区別されます。

GLOBAL アナリティック・ワークスペースの Units メジャーを問い合わせるには、次の手順を実行します。

1. 任意のテキスト・エディタでファイルを開き、例 7-5 に示す SQL スクリプトの本文を入力します。units_query.sql などの名前を付けて保存します。
2. GLOBAL アナリティック・ワークスペースへのアクセス権を持つユーザー名で、SQL*Plus セッションを開きます。
3. 次に示すようなコマンドを使用して、SQL スクリプトを実行します。

```
@units_query
```

例 7-6 に、スクリプトの実行結果を示します。セッションで定義した TS_ROW および TS_TABLE オブジェクトも作成され、これらは他の問合せで使用できます。これらのオブジェクト定義を表示するには、SQL の DESCRIBE 文を使用します。COMMIT を発行すると、オブジェクトが永続的に保存されます。

UNITS_PP または UNITS_PCTCHG_PP には、スタンダード・フォーム・メタデータもアプリケーション・メタデータもありません。

次に、例の説明に対応するコードを示します。

例 7-5 OLAP_TABLE を使用した UNITS メジャーの問合せ

```
CREATE TYPE ts_row AS OBJECT(  
    time_dim          VARCHAR2(4),  
    customer_dim      VARCHAR2(4),  
    product_dim       VARCHAR2(4),  
    channel_dim       VARCHAR2(4),  
    time_name         VARCHAR2(8),  
    customer_name     VARCHAR2(36),  
    product_name      VARCHAR2(36),  
    channel_name      VARCHAR2(12),  
    units             NUMBER(16),  
    units_pp          NUMBER(16),  
    units_pctchg_pp   NUMBER(16));  
/  
CREATE TYPE ts_table AS TABLE OF ts_row;  
/  
SELECT time_name, units, units_pp, units_pctchg_pp  
FROM TABLE(OLAP_TABLE(  
    'global DURATION SESSION',  
    'ts_table',  
    'LIMIT customer_hierlist to 2',  
    'MEASURE units FROM units  
    MEASURE units_pp FROM units_pp  
    MEASURE units_pctchg_pp FROM units_pctchg_pp  
    DIMENSION channel_dim FROM channel WITH  
        HIERARCHY channel_parentrel  
        ATTRIBUTE channel_name FROM channel_long_description  
  
    DIMENSION product_dim FROM product WITH  
        HIERARCHY product_parentrel  
        ATTRIBUTE product_name FROM product_long_description  
    DIMENSION customer_dim FROM customer WITH  
        HIERARCHY customer_parentrel  
        ATTRIBUTE customer_name FROM customer_long_description  
    DIMENSION time_dim FROM time WITH  
        HIERARCHY time_parentrel  
        ATTRIBUTE time_name FROM time_long_description'))  
WHERE units IS NOT NULL AND  
    channel_dim = '1' AND  
    product_dim = '1' AND  
    customer_dim = '21';  
/
```

例 7-6 スクリプトの実行結果

```
TIME_NAM      UNITS      UNITS_PP  UNITS_PCTCHG_PP  
-----
```

Jan-98	11357		
Feb-98	11336	11357	0
Mar-98	11184	11336	-.01
Apr-98	11309	11184	.01
May-98	11489	11309	.02
Jun-98	11423	11489	-.01
Jul-98	11923	11423	.04
Aug-98	12196	11923	.02
	.		
	.		
	.		

TS_ROW オブジェクトの定義

例 7-5 では、TS_ROW オブジェクト（名前の TS は Time Series（時系列）の略）によって、各ディメンションの列、各ディメンションの説明的な名前の列、および各メジャーの列が定義されています。これは、OLAP API または Discoverer で使用される形式ではありませんが、OLAP_TABLE ファンクションを使用して、アプリケーションで必要な任意の形式が作成可能であることを単に示したものです。

メジャーを含む表型はすべて、メジャーの各ディメンションの列も含む必要があります。ディメンションはデータ選択で必要ですが、表示で使用する必要はありません。

OLAP_TABLE ファンクション

例 7-5 では、OLAP_TABLE の引数に、表示するメジャー、そのディメンション、データの意味を示す説明的なディメンション名などの最も基本的な情報を指定しています。また、OLAP_TABLE ファンクションでは、ディメンションの階層構造を定義する親リレーションの名前が必要です。これらのディメンションは、Analytic Workspace Manager のアナリティック・ワークスペース作成ウィザードで作成したため、親リレーションの名前は dimension_PARENTREL になっています。

OLAP_TABLE ファンクションは、複数の SELECT 文に対して、TS_ROW および TS_TABLE を 1 回定義できる SQL セッションを開きます（開いていない場合）。

CUSTOMER ディメンションは 2 つの階層を持ち、LIMIT コマンドは、2 番目の階層の MARKET_SEGMENTS を選択します。SHIPMENTS は、CUSTOMER_HIERLIST 階層ディメンションの最初の階層で、これがデフォルトです。その他のディメンションの階層は 1 つのみです。したがって、hierlist ディメンションを制限する必要はありません。

TS_ROW で定義したすべての列のソース・データは、制限マップで識別されます。ソースはすべて、アナリティック・ワークスペースのオブジェクトです。ただし、UNITS_LAGPCT_PP を除きます。これは、AW_EXPR 引数を指定することにより、MEASURE 句で計算されます。

SELECT 文

例 7-5 の SELECT 文では、データベース内の物理的な表とまったく同様に、TS_TABLE から必要な列と行を識別します。このような特定選択では、TIME を除くすべてのディメンションが、WHERE 句によって単一の値に制限され、TIME の long description のみが結果セットにラベル付けされます。

OLAP_TABLE を使用した OLAP API 用のビューの作成

例 7-7 に、OLAP_TABLE を使用して、アナリティック・ワークスペースのデータを OLAP API および BI Beans で使用可能にする方法を示します。この処理では次の手順を実行します。

1. OLAP API の要件を満たすビューを作成します。
2. OLAP API でビューの問合せを実行できるよう OLAP カタログ・メタデータを定義します。

この例では、UNITS、UNITS_PP および UNITS_PCTCHG_PP のビューを作成します。

SQL スクリプトの作成および実行

OLAP API 用のビューを作成するには、次の手順を実行します。

1. 任意のテキスト・エディタでファイルを開き、例 7-7 に示す SQL スクリプトの本文を入力します。ts_views.sql などの名前を付けて保存します。
2. GLOBAL アナリティック・ワークスペースへのアクセス権を持つユーザー名で、SQL*Plus セッションを開きます。
3. 次に示すようなコマンドを使用して、SQL スクリプトを実行します。

```
@ts_views
```

4. データベースへの変更をコミットします。
5. ビューに対して SELECT コマンドを発行して、ビューが適切に定義されたことを確認します。適切に定義されていない場合、エラーが発生します。

例 7-7 OLAP API 用のビューの作成

```
SET ECHO ON
SET SERVEROUT ON
DROP TYPE ts_table_1;
DROP TYPE ts_table_2;
DROP TYPE ts_row;

CREATE TYPE ts_row AS OBJECT(
    channel_et          VARCHAR2(4),
    channel_gid         NUMBER(2),
```

```

product_et          VARCHAR2(4),
product_gid         NUMBER(2),
customer_et         VARCHAR2(4),
customer_gid        NUMBER(2),
time_et            VARCHAR2(8),
time_gid           NUMBER(2),
units              NUMBER(16),
units_pp           NUMBER(16),
units_pctchg_pp     NUMBER(8,2),
r2c                RAW(32));
/
CREATE TYPE ts_table_1 AS TABLE OF ts_row;
/
CREATE OR REPLACE VIEW ts_view_1 AS
SELECT channel_et, channel_gid, product_et, product_gid,
       customer_et, customer_gid, time_et, time_gid,
       units, units_pp, units_pctchg_pp, r2c
FROM TABLE(CAST(OLAP_TABLE(
'global DURATION SESSION',
'ts_table_1',
'LIMIT customer_hierlist to 1',
'MEASURE units FROM units
MEASURE units_pp FROM units_pp
MEASURE units_pctchg_pp FROM AW_EXPR lagpct(units, 1, time, levelrel time_levelrel)
ROW2CELL r2c
DIMENSION channel_et FROM channel WITH
    HIERARCHY channel_parentrel
    INHIERARCHY channel_inhier
    GID channel_gid FROM channel_gid
DIMENSION product_et FROM product WITH
    HIERARCHY product_parentrel
    INHIERARCHY product_inhier
    GID product_gid FROM product_gid
DIMENSION customer_et FROM customer WITH
    HIERARCHY customer_parentrel
    INHIERARCHY customer_inhier
    GID customer_gid from customer_gid
DIMENSION time_et FROM time WITH
    HIERARCHY time_parentrel
    INHIERARCHY time_inhier
    GID time_gid FROM time_gid')
AS ts_table_1))
WHERE units IS NOT NULL;
/
CREATE TYPE ts_table_2 AS TABLE OF ts_row;
/
CREATE OR REPLACE VIEW ts_view_2 AS

```

```
SELECT channel_et, channel_gid, product_et, product_gid,
       customer_et, customer_gid, time_et, time_gid,
       units, units_pp, units_pctchg_pp, r2c
FROM TABLE(CAST(OLAP_TABLE(
  'global DURATION SESSION',
  'ts_table_2',
  'LIMIT customer_hierlist to 2',
  'MEASURE units FROM units
   MEASURE units_pp FROM units_pp
   MEASURE units_pctchg_pp FROM AW_EXPR lagpct(units, 1, time, levelrel time_levelrel)
  ROW2CELL r2c
  DIMENSION channel_et FROM channel WITH
    HIERARCHY channel_parentrel
    INHIERARCHY channel_inhier
    GID channel_gid FROM channel_gid
  DIMENSION product_et FROM product WITH
    HIERARCHY product_parentrel
    INHIERARCHY product_inhier
    GID product_gid FROM product_gid
  DIMENSION customer_et FROM customer WITH
    HIERARCHY customer_parentrel
    INHIERARCHY customer_inhier
    GID customer_gid from customer_gid
  DIMENSION time_et FROM time WITH
    HIERARCHY time_parentrel
    INHIERARCHY time_inhier
    GID time_gid FROM time_gid')
  AS ts_table_2))
WHERE units IS NOT NULL;
```

サンプル・スクリプトについて

例 7-7 では、ファクト表について OLAP API の要件を満たす TS_ROW という型を定義しています。

- 各ディメンションは、すべての階層レベルのメンバーに対して埋込み合計列を 1 つ持ちます。列の名前は、*dimension_ET* となり、OLAP API イネーブラで生成されるビューに準拠します。
- 各ディメンションは、そのグルーピング ID の列を持ちます。列の名前は、*dimension_GID* となり、OLAP API イネーブラで生成されるビューに準拠します。
- OLAP_EXPRESSION ファンクションで使用するために、ROW2CELL 列を定義しています。

TS_ROW を使用して、各 Customer 階層に 1 つずつ、2 つの表が定義されています。有効なキューブを作成するために、OLAP カタログでは、階層の各組合せに対してビューが必要となります。各ディメンションについては、表によって次の項目が識別されます。

- HIERARCHY リレーション。各メンバーの親を識別することにより、ディメンション・メンバー間の階層関係を定義します。
- INHIERARCHY 変数。選択した階層にディメンション・メンバーがあるかどうかを識別します。
- GID 変数。各ディメンション・メンバーのグルーピング ID を識別します。

これらのオブジェクトは、アナリティック・ワークスペース作成ウィザードで作成したものです。ビュー内の列にマップされるのは GID 変数のみです。

ワークスペース・ビューの OLAP カタログ・メタデータの定義

アナリティック・ワークスペースのビューの OLAP カタログ・メタデータを定義するには、CWM2 書込み API を使用する必要があります。これにより、Analytic Workspace Manager の OLAP カタログ・ビューで CWM2 メタデータを表示できるようになります。これは、OLAP カタログ・ビューを直接問い合せても表示できます。Oracle Enterprise Manager を使用した場合、CWM2 メタデータを定義することも表示することもできません。

既存の Units キューブに、新しいメジャー（UNITS_PP および UNITS_PCTCHG_PP）を追加できます。ただし、例 7-8 では、事前に定義したディメンションを使用して新しいキューブを作成する方法を示しています。この例では、新しいメジャー・フォルダも作成します。

新しいメジャーの OLAP カタログ・メタデータを作成するには、次の手順を実行します。

1. 任意のテキスト・エディタでファイルを開き、例 7-8 に示す SQL スクリプトの本文を入力します。ts_cwm.sql などの名前を付けて保存します。

CWM2 API の構文および使用上の注意の詳細は、『Oracle OLAP リファレンス』を参照してください。

2. GLOBAL アナリティック・ワークスペースへのアクセス権を持つユーザー名で SQL*Plus セッションを開き、次のコマンドを発行します。

```
SET ECHO ON
SET LINESIZE 135
SET PAGESIZE 50
SET SERVEROUTPUT ON FORMAT WRAPPED SIZE 1000000
EXECUTE CWM2_OLAP_MANAGER.SET_ECHO_ON;
```

この設定により、エラー・メッセージを確認したり、スクリプトで実行される検証プログラムからの詳細なレポートを表示したりできます。メタデータはデータベースにコミットする前に検証することが重要です。

3. 次に示すようなコマンドを使用して、SQL スクリプトを実行します。

```
@ts_cwm
```

注意： SQL*Plus の最大バッファ・サイズを検証メッセージが超過した場合、CWM2_OLAP_MANAGER.BEGIN_LOG を使用すると、ログ・ファイルに検証メッセージをリダイレクトできます。

4. エラーが発生した場合は、次の手順を実行します。
 - a. ROLLBACK コマンドを発行します。
 - b. スクリプトのエラーを修正します。
 - c. スクリプトを再実行します。
5. 次のように、メタデータを OLAP API および BI Beans 用の特別なビューにコピーします。

```
EXECUTE CWM2_OLAP_METADATA_REFRESH.MR_REFRESH();
```

このプロシージャでは、COMMIT を発行します。

OLAP カタログで定義したメジャーは、スタンダード・フォームのメジャーと同様に OLAP API および BI Beans アプリケーションで使用できるようになります。図 7-3 に、BI Beans アプリケーションで発行した問合せの結果セットを示します。

図 7-3 BI Beans サンプル・アプリケーションを使用して問い合わせた新しいメジャー

The screenshot shows the BI Beans Sample - Analyzer application window. The main area displays a data table with the following columns: Units Sold, Units Prior Period, and Units Pct Chg PP. The table is filtered by 'All Channels' and shows data for 2001, Q1-01, Jan-01, Feb-01, Mar-01, Q2-01, Q3-01, and Q4-01. The status bar at the bottom indicates 'Connected'.

	Units Sold	Units Prior Period	Units Pct Chg PP
2001	415,392	364,233	0.14
Q1-01	96,383	97,184	-0.01
Jan-01	32,572	32,233	0.01
Feb-01	31,987	32,572	-0.02
Mar-01	31,824	31,987	-0.01
Q2-01	97,346	96,383	0.01
Q3-01	105,704	97,346	0.09
Q4-01	115,959	105,704	0.10

例 7-8 GLOBAL メジャーの OLAP カタログ・メタデータを作成するスクリプト

```

BEGIN
-- Define TS_CUBE cube with predefined dimensions
CWM2_OLAP_CUBE.CREATE_CUBE('GLOBAL_AW', 'TS_CUBE', 'TS Cube', 'TS Cube', 'Units Time Series Cube');
CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBAL_AW', 'TS_CUBE', 'GLOBAL_AW', 'CHANNEL');
CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBAL_AW', 'TS_CUBE', 'GLOBAL_AW', 'PRODUCT');
CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBAL_AW', 'TS_CUBE', 'GLOBAL_AW', 'CUSTOMER');
CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBAL_AW', 'TS_CUBE', 'GLOBAL_AW', 'Time');
CWM2_OLAP_MEASURE.CREATE_MEASURE('GLOBAL_AW', 'TS_CUBE', 'UNITS_PP', 'Units PP',
    'Units Prior Period', 'Units Sold in Prior Period');
CWM2_OLAP_MEASURE.CREATE_MEASURE('GLOBAL_AW', 'TS_CUBE', 'UNITS_PCTCHG_PP', 'Units PctChgPP',
    'Units Pct Chg PP', 'Percent Difference in Units Sold From Prior Period');

-- Map TS_VIEW_1 view to metadata cube TS_CUBE
CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_LEVELKEY('GLOBAL_AW', 'TS_CUBE', 'GLOBAL_AW', 'TS_VIEW_1', 'ET',
    'DIM:GLOBAL_AW.CHANNEL/HIER:CHANNEL_ROLLUP/GID:CHANNEL_GID/LVL:CHANNEL/COL:CHANNEL_ET;
    DIM:GLOBAL_AW.CUSTOMER/HIER:SHIPMENTS/GID:CUSTOMER_GID/LVL:SHIP_TO/COL:CUSTOMER_ET;
    DIM:GLOBAL_AW.PRODUCT/HIER:PRODUCT_ROLLUP/GID:PRODUCT_GID/LVL:ITEM/COL:PRODUCT_ET;
    DIM:GLOBAL_AW.Time/HIER:Calendar/GID:TIME_GID/LVL:Month/COL:TIME_ET');

CWM2_OLAP_TABLE_MAP.ADD_AWVIEW('GLOBAL_AW', 'TS_VIEW_1', 'r2c');

CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_MEASURE('GLOBAL_AW', 'TS_CUBE', 'UNITS_PP', 'GLOBAL_AW', 'TS_VIEW_1',
    'UNITS_PP',
    'DIM:GLOBAL_AW.CHANNEL/HIER:CHANNEL_ROLLUP/GID:CHANNEL_GID/LVL:CHANNEL/COL:CHANNEL_ET;
    DIM:GLOBAL_AW.CUSTOMER/HIER:SHIPMENTS/GID:CUSTOMER_GID/LVL:SHIP_TO/COL:CUSTOMER_ET;
    DIM:GLOBAL_AW.PRODUCT/HIER:PRODUCT_ROLLUP/GID:PRODUCT_GID/LVL:ITEM/COL:PRODUCT_ET;
    DIM:GLOBAL_AW.TIME/HIER:Calendar/GID:TIME_GID/LVL:Month/COL:TIME_ET');

CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_MEASURE('GLOBAL_AW', 'TS_CUBE', 'UNITS_PCTCHG_PP', 'GLOBAL_AW',
    'TS_VIEW_1', 'UNITS_PCTCHG_PP',
    'DIM:GLOBAL_AW.CHANNEL/HIER:CHANNEL_ROLLUP/GID:CHANNEL_GID/LVL:CHANNEL/COL:CHANNEL_ET;
    DIM:GLOBAL_AW.CUSTOMER/HIER:SHIPMENTS/GID:CUSTOMER_GID/LVL:SHIP_TO/COL:CUSTOMER_ET;
    DIM:GLOBAL_AW.PRODUCT/HIER:PRODUCT_ROLLUP/GID:PRODUCT_GID/LVL:ITEM/COL:PRODUCT_ET;
    DIM:GLOBAL_AW.TIME/HIER:Calendar/GID:TIME_GID/LVL:Month/COL:TIME_ET');

CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_LEVELKEY('GLOBAL_AW', 'TS_CUBE', 'GLOBAL_AW', 'TS_VIEW_2', 'ET',
    'DIM:GLOBAL_AW.CHANNEL/HIER:CHANNEL_ROLLUP/GID:CHANNEL_GID/LVL:CHANNEL/COL:CHANNEL_ET;
    DIM:GLOBAL_AW.CUSTOMER/HIER:MARKET_SEGMENT/GID:CUSTOMER_GID/LVL:SHIP_TO/COL:CUSTOMER_ET;
    DIM:GLOBAL_AW.PRODUCT/HIER:PRODUCT_ROLLUP/GID:PRODUCT_GID/LVL:ITEM/COL:PRODUCT_ET;
    DIM:GLOBAL_AW.TIME/HIER:Calendar/GID:TIME_GID/LVL:Month/COL:TIME_ET');

CWM2_OLAP_TABLE_MAP.ADD_AWVIEW('GLOBAL_AW', 'TS_VIEW_2', 'r2c');

-- Map TS_VIEW_2 view to metadata cube TS_CUBE
CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_MEASURE('GLOBAL_AW', 'TS_CUBE', 'UNITS_PP', 'GLOBAL_AW', 'TS_VIEW_2',
    'UNITS_PP',

```

事例 : OLAP_TABLE を使用した Global カスタム・メジャーの作成

```
'DIM:GLOBAL_AW.CHANNEL/HIER:CHANNEL_ROLLUP/GID:CHANNEL_GID/LVL:CHANNEL/COL:CHANNEL_ET;
DIM:GLOBAL_AW.CUSTOMER/HIER:MARKET_SEGMENT/GID:CUSTOMER_GID/LVL:SHIP_TO/COL:CUSTOMER_ET;
DIM:GLOBAL_AW.PRODUCT/HIER:PRODUCT_ROLLUP/GID:PRODUCT_GID/LVL:ITEM/COL:PRODUCT_ET;
DIM:GLOBAL_AW.TIME/HIER:Calendar/GID:TIME_GID/LVL:Month/COL:TIME_ET;');

CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_MEASURE('GLOBAL_AW', 'TS_CUBE', 'UNITS_PCTCHG_PP', 'GLOBAL_AW',
'TS_VIEW_2', 'UNITS_PCTCHG_PP',
'DIM:GLOBAL_AW.CHANNEL/HIER:CHANNEL_ROLLUP/GID:CHANNEL_GID/LVL:CHANNEL/COL:CHANNEL_ET;
DIM:GLOBAL_AW.CUSTOMER/HIER:MARKET_SEGMENT/GID:CUSTOMER_GID/LVL:SHIP_TO/COL:CUSTOMER_ET;
DIM:GLOBAL_AW.PRODUCT/HIER:PRODUCT_ROLLUP/GID:PRODUCT_GID/LVL:ITEM/COL:PRODUCT_ET;
DIM:GLOBAL_AW.TIME/HIER:Calendar/GID:TIME_GID/LVL:Month/COL:TIME_ET;');

-- Validate the cube metadata
CWM2_OLAP_VALIDATE.VALIDATE_CUBE('GLOBAL_AW', 'TS_CUBE', 'OLAP_API');

-- Create a measure folder
CWM2_OLAP_CATALOG.CREATE_CATALOG('GLOBAL_ANALYTIC_CAT', 'Global Analytic Measures');
CWM2_OLAP_CATALOG.ADD_CATALOG_ENTITY('GLOBAL_ANALYTIC_CAT', 'GLOBAL_AW', 'TS_CUBE', 'UNITS');
CWM2_OLAP_CATALOG.ADD_CATALOG_ENTITY('GLOBAL_ANALYTIC_CAT', 'GLOBAL_AW', 'TS_CUBE', 'UNITS_PP');
CWM2_OLAP_CATALOG.ADD_CATALOG_ENTITY('GLOBAL_ANALYTIC_CAT', 'GLOBAL_AW', 'TS_CUBE',
'UNITS_PCTCHG_PP');
--COMMIT;
end;
/
```

スタンダード・フォームのアナリティック・ワークスペースの詳細

この章では、スタンダード・フォームのアナリティック・ワークスペースで作成されるオブジェクトについて説明します。この章は、独自のアナリティック・ワークスペースのガイドとして利用できます。また、Analytic Workspace Manager のオブジェクト・ビューを開くと、この章で説明するオブジェクトのプロパティ・シートを確認できます。

この章では、次の項目について説明します。

- [OLAP ツールを使用して作成されるワークスペースについて](#)
- [スタンダード・フォームのディメンション](#)
- [スタンダード・フォームの階層](#)
- [スタンダード・フォームのレベル](#)
- [スタンダード・フォームの属性](#)
- [スタンダード・フォームのメジャー](#)
- [スタンダード・フォームのキューブ](#)
- [スタンダード・フォームのカタログ](#)
- [OLAP API イネーブラのカタログ](#)
- [AWCREATE カタログ](#)

参照： データベース・スタンダード・フォームの仕様の詳細は、[付録C](#)を参照してください。

OLAP ツールを使用して作成されるワークスペースについて

第6章で説明したように、アナリティック・ワークスペースを作成する方法は複数あります。どの方法でアナリティック・ワークスペースを作成しても、同じ基本特性を持ちます。

これらの特性には、[データベース・スタンダード・フォーム](#)規則との準拠も含まれていません。

データベース・スタンダード・フォームについて

様々な方法でリレーショナル・スキーマを設定できるのと同様に、アナリティック・ワークスペースの設計方法は、アプリケーション開発者の数と同じくらいあります。ただし、アナリティック・ワークスペースに対して実行するアプリケーションを作成する場合、アプリケーションでは、ワークスペース内で特定のオブジェクトを検出してルールを識別できるよう、ある特定の設計が必要となります。Analytic Workspace Manager で使用可能なツールの設計を[データベース・スタンダード・フォーム](#)と呼びます。

Analytic Workspace Manager および現行バージョンのツールは、データベース・スタンダード・フォームのアナリティック・ワークスペースでのみ使用できます。データベース・スタンダード・フォーム（単に、スタンダード・フォーム）では、次の要件を規定しています。

- 特定のオブジェクトがアナリティック・ワークスペースに存在している必要があります。これらのオブジェクトおよびプロパティは、集計、データのリフレッシュおよびアプリケーションの有効化などのタスクを実行する、Analytic Workspace Manager のツールで使用されます。DBMS_AW_UTILITIES など一部の PL/SQL パッケージと同様に、アクティブ・カタログもデータベース・スタンダード・フォームに依存します（7-1 ページの「[SQL アクセスの概要](#)」を参照）。
- OLAP DML プロパティ（AW\$ で始まる）は、これらのオブジェクト上で定義する必要があります。プロパティ値はオブジェクトのメタデータであり、アナリティック・ワークスペースにある他のオブジェクトとの関係を識別します。
- オブジェクトは、ワークスペース・カタログに登録する必要があります。OLAP ツールでは、これらのメタデータ・カタログを問い合せて、アナリティック・ワークスペースにおける論理キューブ、メジャーおよびディメンションのインスタンス化方法に関する情報を取得します。Analytic Workspace Manager のツールを使用してオブジェクトを定義する際、ツールではカタログのメンテナンスも行います。ただし、手動でオブジェクトを定義する場合は、このマニュアルの他の章でも説明しているように、ツールで新しいオブジェクトを認識できるよう、プロパティおよびカタログをメンテナンスする必要があります。

Analytic Workspace Manager のアナリティック・ワークスペース作成ウィザードを使用すると、スタンダード・フォームのアナリティック・ワークスペースが作成されます。オブジェクト・ビューを使用してワークスペース・オブジェクトを参照することで、スタンダード・フォームを十分に理解することができます。

スタンダード・フォームで必要なすべてのオブジェクトが、アナリティック・ワークスペースのツールで現在使用されているわけではありません。このようなオブジェクトについては、この章で説明しておらず、現時点では無視しても構いません。

参照： データベース・スタンダード・フォームの規則の詳細は、[付録 C](#) を参照してください。

アナリティック・ワークスペースの追加要件

Analytic Workspace Manager の一部のツールでは、追加のプロパティおよびオブジェクトが必要になります。これらは、アナリティック・ワークスペースで定義します。

一部のプロパティ値は、DBMS_AWM プロシージャの作成パラメータを識別します。DBMS_AWM を直接実行してアナリティック・ワークスペースを作成した場合、これらの値はすぐに確認できます。Analytic Workspace Manager または Oracle Warehouse Builder を使用した場合、DBMS_AWM へのコールが生成され、作成方法を選択できます。

スタンダード・フォームでは、ワークスペース・オブジェクトのネーミング規則は指定しません。ただし、DBMS_AWM によって作成されるオブジェクトには標準名が付けられます。通常、この名前はアナリティック・ワークスペース内のオブジェクトのロールを識別します。この章では、AWSROLE プロパティの値に基づいてオブジェクトを識別し、DBMS_AWM で付けられた標準名を識別します。アナリティック・ワークスペースを作成する場合、これらの名前に接頭辞を追加するかどうかを選択できます。

この章では、「スタンダード・フォーム」という用語を広い意味で使用しています。つまり、規則および DBMS_AWM による規則の実装の両方の意味で使用しています。

スタンダード・フォームのディメンション

一般的にキューブのディメンションは本質的に階層的であるため、ディメンションは複数のレベルおよび階層を持ちます。アナリティック・ワークスペースのディメンションは、すべてのレベルでメンバーを含んでいるため、**埋込み合計**ディメンションと呼ばれることがよくあり、集計データを含むメジャーの定義に使用されます。ディメンション・メンバーは、リレーショナル・ディメンション表の複数のレベル列から取得されます。

埋込み合計ディメンションは、ディメンション・オブジェクトの他に、少なくとも 1 つのレベルと 1 つの階層を持ちます。フラット・ディメンションの場合、レベルと階層は必要ありません。

ディメンションはすべて、デフォルトの順序属性を持ちます (8-16 ページの「**スタンダード・フォームの属性**」を参照)。時間属性には、end date 属性と time span 属性を指定する必要があります。

ディメンションの詳細は、C-14 ページの「**実装クラスのオブジェクト**」を参照してください。

Dimdef ディメンション

アナリティック・ワークスペースの *dimdef* ディメンション (キューブで使用されるディメンション) は、メタデータで定義されている名前 (TIME または PRODUCT など) を持ち、作成時に指定した接頭辞が付いている場合もあります。ディメンション・メンバーをロードする前にディメンションを再定義しないかぎり、ディメンションのデータ型は TEXT です。ディメンション・メンバーの名前には、ソース値にレベル名の接頭辞が追加されている場合もあります。

アナリティック・ワークスペース・ディメンションの内容

アナリティック・ワークスペースのディメンション・メンバーは、リレーショナル・ディメンション表のメンバーとまったく同じ場合もあれば、レベル名の接頭辞が付いている場合もあります。作成時に接頭辞を付けるかどうかはオプションです。[例 8-1](#) では、作成時に接頭辞を指定した場合に、Global の PRODUCT ディメンション・メンバーがどのように表示されるかを示します。Global スター・スキーマによってサロゲート・キーが提供されるため、実際にはレベル間でのディメンション・メンバーの一意性を保証する目的で接頭辞を付ける必要はありません。

ディメンション・メンバーはすべて、ロード・プロセス時に格納されます。Time ディメンションの場合、メンバーはレベル別およびレベル内の終了日別にソートされます。この順序は、ディメンション内の期間の相対位置に基づいた時系列分析をサポートするために必要です。その他のディメンションは、レベル別およびレベル内のディメンション・メンバー別に英数字順にソートされます。デフォルトの順序属性では、ディメンション・メンバーがアナリティック・ワークスペースにロードされた元の順序を識別します。

例 8-1 レベル名の接頭辞付きの Global の Product

```
LIMIT product TO product_levelrel EQ 'ITEM'
LIMIT product KEEP FIRST 3
LIMIT product ADD ANCESTORS USING product_parentrel
REPORT W 20 product

PRODUCT
-----
ITEM.13
ITEM.14
ITEM.15
FAMILY.4
CLASS.2
TOTAL_PRODUCT.1
```

アナリティック・ワークスペースの Dimdef ディメンションのプロパティ

[表 8-1](#) に、*dimdef* ディメンションの OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、[付録 C](#) を参照してください。

表 8-1 Dimdef ディメンションのプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION
AW\$CREATEDBY	ディメンションの作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、ディメンションが AWCREATE プログラムで作成されたことを示します。

表 8-1 Dimdef ディメンションのプロパティ

プロパティ	値
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にディメンションへのアクセスが行われた日付と時刻。
AW\$LOGICAL_NAME	OLAP カタログのソース名。
AW\$PARENT_NAME	NA
AW\$ROLE	DIMDEF
AW\$STATE	ACTIVE
AW\$TYPE	Time ディメンションの場合は Time、それ以外の場合は NA。
DESCRIPTION	ディメンションのオプションの説明。
LOAD_TYPE	最後のリフレッシュに実行されたロード・タイプ。値は DBMS_AWM.CREATE_AWDIMLOAD_SPEC で指定した FULL_LOAD_ADDITIONS_ONLY または FULL_LOAD のいずれかです。これは、AWCREATE プログラムで使用されます。
SOURCE_NAME	OLAP カタログのソースのディメンション。これは、AWCREATE プログラムおよびアクティブ・カタログで使用されます。
SOURCE_OWNER	OLAP カタログのソース・メタデータの所有者。これは、AWCREATE プログラムおよびアクティブ・カタログで使用されます。
UNIQUE_RDBMS_KEY	ソース・ディメンション表でレベル間の一意キーが提供された場合は YES、一意性を保証するためにレベル名の接頭辞がキーに付けられた場合は NO。値は、DBMS_AWM.SET_AWDIMLOAD_SPEC_PARAMETER で指定します。これは、AWCREATE プログラムで使用されます。
DISPLAY_NAME	OLAP カタログのソースの表示名または DBMS_AWM.SET_AWDIMLOAD_SPEC_PARAMETER の設定。これは、AWCREATE プログラムで使用されます。
P_DISPLAY_NAME	OLAP カタログの複数表示名または DBMS_AWM.SET_AWDIMLOAD_SPEC_PARAMETER の設定。

ディメンションのスタンダード・フォーム・メタデータ

ディメンションのスタンダード・フォーム・メタデータは、次のオブジェクトに格納されます。

- ALL_DIMENSIONS ディメンション
- ALL_DESCRIPTIONS 変数

- AW_NAMES 変数
- DIM_LEVELS 値セット

ALL_DIMENSIONS ディメンション

ALL_DIMENSIONS ディメンションは、次の形式ですべてのディメンションの名前を格納します。

```
workspace.dimension.DIMENSION
```

例: GLOBAL_AW.PRODUCT.DIMENSION

ALL_DIMENSIONS は、ALL_OBJECTS CONCAT ディメンションのベース・ディメンションです。ALL_OBJECTS は、ALL_DESCRIPTIONS および AW_NAMES をディメンション化するため、これらのカタログは各メジャーのエントリを持ちます。

ディメンションの ALL_DESCRIPTIONS 変数

ALL_DESCRIPTIONS 変数は、ディメンションの簡単な説明、詳細な説明および複数説明を格納します。すべてのオブジェクトはメタデータから取得した簡単な説明を持ちますが、詳細な説明と複数説明は持たない場合もあります。

ディメンションの AW_NAMES 変数

AW_NAMES メジャーは、ワークスペース・ディメンション・オブジェクトの完全修飾された名前を次の形式で提供します。

```
schema.workspace!dimension
```

例: GLOBAL_AW.GLOBAL!PRODUCT

DIM_LEVELS 値セット

DIM_LEVELS 値セットは、各ディメンションに定義されているレベルを識別します。

スタンダード・フォームの階層

ディメンション階層をサポートするオブジェクトは次のとおりです。

- *Hierlist* ディメンション
- *Member_parentrel* リレーション
- *Member_gid* 変数
- *Member_inhier* 変数

member_parentrel リレーション、*member_gid* 変数および *member_inhier* 変数の値は階層ごとに異なる場合があるので、これらのオブジェクトの定義には、*hierlist* ディメンションを使用します。

階層の詳細は、C-14 ページの「[実装クラスのオブジェクト](#)」および C-34 ページの「[機能クラスのオブジェクト](#)」を参照してください。

Hierlist ディメンション

hierlist ディメンションは、特定のディメンションに定義されている階層の名前を格納します。階層の名前は、OLAP カタログから取得されます。通常、このテキスト・ディメンションは *dimdef_HIERLIST* という名前を持ちます。

Hierlist ディメンションの内容

[例 8-2](#) に、GLOBAL アナリティック・ワークスペースの CUSTOMER_HIERLIST の内容を示します。

例 8-2 GLOBAL の CUSTOMER の Hierlist ディメンション

```
REPORT W 20 customer_hierlist
```

```
CUSTOMER_HIERLIST
-----
SHIPMENTS
MARKET_SEGMENT
```

Hierlist ディメンションのプロパティ

[表 8-2](#) に、*hierlist* ディメンションの OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、[付録 C](#) を参照してください。

表 8-2 Hierlist ディメンションのプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION
AW\$CREATEDBY	ディメンションの作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、ディメンションが AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にディメンションへのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	Dimdef ディメンション。
AW\$ROLE	HIERLIST
AW\$STATE	CREATED

Member_Parentrel リレーション

member_parentrel リレーションは、各メンバーの親を識別することにより、ディメンション・メンバー間の階層関係を定義します。このリレーションでは、ディメンションに必要な階層のサポートを提供します。この情報は、リレーションル・ディメンション表から取得されます。親リレーションには、*dimdef_PARENTREL* という名前が付けられます。

Member_Parentrel リレーションの内容

member_parentrel リレーションは、有効な値のみがディメンション・メンバーであるセルフ・リレーションの一種です。例 8-3 に、GLOBAL アナリティック・ワークスペースの CHANNEL ディメンションの *member_parentrel* リレーションを示します。リレーションでは 2 つのレベルの階層を定義します。この中で、1 は、2、3 および 4 の親です。

例 8-3 GLOBAL の CHANNEL の Member_Parentrel リレーション

```
REPORT DOWN channel W 20 channel_parentrel

                                -CHANNEL_PARENTREL--
                                --CHANNEL_HIERLIST--
CHANNEL                        CHANNEL_ROLLUP
-----
1                             NA
2                             1
3                             1
4                             1
```

Member_Parentrel リレーションのプロパティ

表 8-3 に、member_parentrel リレーションの OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-3 Member_Parentrel リレーションのプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION
AW\$CREATEDBY	リレーションの作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、リレーションが AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にリレーションへのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	Dimdef デイメンション。
AW\$ROLE	MEMBER_PARENTREL
AW\$STATE	CREATED

Member_Gid 変数

Member_gid 変数を使用すると、OLAP API のビューのパフォーマンスが向上します。この整数は、各デイメンション・メンバーの階層の深さを識別します。この情報は、OLAP DML の GROUPINGID コマンドで生成されます。member_gid 変数の標準名は、dimdef_gid です。

Member_Gid 変数の内容

例 8-4 に、GLOBAL アナリティック・ワークスペースの CHANNEL デイメンションの member_gid 変数を示します。この例では、チャンネル 2、3、4 はベース・レベル（0）で、チャンネル 1 は 1 レベルの深さ（1）にあります。

例 8-4 GLOBAL の CHANNEL の Member_Gid

```
REPORT DOWN channel W 20 channel_gid

          ----CHANNEL_GID-----
          --CHANNEL_HIERLIST--
CHANNEL   CHANNEL_ROLLUP
-----
1                1
2                0
3                0
4                0
```

Member_Gid 変数のプロパティ

表 8-4 に、*member_gid* 変数の OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-4 Member_Gid 変数のプロパティ

プロパティ	値
AW\$CLASS	FEATURES
AW\$CREATEDBY	変数の作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、変数が AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後に変数へのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	<i>Dimdef</i> デイメンション。
AW\$ROLE	MEMBER_GID
AW\$STATE	CREATED

Member_Inhier 変数

Member_inhier 変数を使用すると、OLAP API のビューのパフォーマンスが向上します。このブール変数では、特定の階層に含まれているレベルにデイメンション・メンバーが属しているかどうかを識別します。情報は OLAP カタログ・メタデータから取得され、通常は複数の階層を持つデイメンションでのみ使用します。*member_inhier* 変数の標準名は、*dimension__INHIER* です。

Member_Inhier 変数の内容

例 8-5 に、GLOBAL アナリティック・ワークスペースの CUSTOMER デイメンションの *member_inhier* 変数の内容を示します。YES はデイメンション・メンバーが階層内にあることを示し、NA は階層内にないことを示します。

例 8-5 GLOBAL の CUSTOMER の Member_Inhier 変数

```
LIMIT customer TO customer_levelrel EQ 'SHIP_TO' "Select base-level members
LIMIT customer KEEP FIRST 1 "Keep just the first one
LIMIT customer ADD ANCESTORS USING customer_parentrel "Add its ancestors
REPORT DOWN customer W 15 customer_inhier

-----CUSTOMER_INHIER-----
-----CUSTOMER_HIERLIST-----
CUSTOMER      SHIPMENTS      MARKET_SEGMENT
-----
```

46	yes	yes
21	yes	NA
22	NA	yes
10	yes	NA
5	NA	yes
1	yes	NA
7	NA	yes

Member_Inhier 変数のプロパティ

表 8-5 に、*member_inhier* 変数の OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-5 Member_Inhier 変数のプロパティ

プロパティ	値
AW\$CLASS	FEATURES
AW\$CREATEDBY	変数の作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、変数が AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後に変数へのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	Dimdef デイメンション。
AW\$ROLE	MEMBER_INHIER
AW\$STATE	CREATED

階層のスタンダード・フォーム・メタデータ

階層のスタンダード・フォーム・メタデータは、次のオブジェクトに格納されます。

- ALL_HIERARCHIES デイメンション
- ALL_DESCRIPTIONS 変数
- DIM_HIERARCHIES 値セット
- DEFAULT_HIER リレーション

ALL_HIERARCHIES デイメンション

ALL_HIERARCHIES デイメンションは、次の形式ですべての階層の名前を格納します。

workspace.dimension.hierarchy.HIERARCHY

例: GLOBAL_AW.CUSTOMER.SHIPMENTS.HIERARCHY

ALL_HIERARCHIES は、ALL_OBJECTS CONCAT ディメンションのベース・ディメンションです。ALL_OBJECTS は、ALL_DESCRIPTIONS および AW_NAMES をディメンション化します。ALL_DESCRIPTIONS は階層の値を提供しますが、AW_NAMES は提供しません。

階層の ALL_DESCRIPTIONS 変数

ALL_DESCRIPTIONS 変数は、階層の簡単な説明、詳細な説明および複数説明を格納します。すべてのオブジェクトはメタデータから取得した簡単な説明を持ちますが、詳細な説明と複数説明は持たない場合もあります。

DIM_HIERARCHIES 値セット

DIM_HIERARCHIES 値セットは、各ディメンションに定義されている階層を識別します。

DEFAULT_HIER リレーション

DEFAULT_HIER リレーションは、各ディメンションのデフォルトの階層を識別します。

スタンダード・フォームのレベル

レベルは、ディメンション階層の基礎となるものです。レベルは、1 つ以上の階層に属します。レベル定義をサポートするオブジェクトは次のとおりです。

- *Levellist* ディメンション
- *Member_levelrel* リレーション
- *Member_familyrel* リレーション

レベルの詳細は、C-14 ページの「[実装クラスのオブジェクト](#)」および C-34 ページの「[機能クラスのオブジェクト](#)」を参照してください。

Levellist ディメンション

levellist ディメンションは、特定のディメンションに定義されているすべての階層のすべてのレベルの名前を格納します。情報は、OLAP カタログから取得されます。通常、このテキスト・ディメンションは、*dimdef_LEVELLIST* という名前を持ちます。

Levellist ディメンションの内容

[例 8-6](#) に、GLOBAL の CUSTOMER の *levellist* ディメンションを示します。このディメンションには、SHIPMENTS 階層と MARKET_SEGMENT 階層の両方のレベルが含まれています。

例 8-6 GLOBAL の CUSTOMER の Levellist ディメンション

```
REPORT W 20 customer_levellist
```

CUSTOMER_LEVELLIST

TOTAL_MARKET
MARKET_SEGMENT
ACCOUNT
ALL_CUSTOMERS
REGION
WAREHOUSE
SHIP_TO

Levellist ディメンションのプロパティ

表 8-6 に、*levellist* ディメンションのプロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-6 Levellist ディメンションのプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION
AW\$CREATEDBY	レベルの作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、レベルが AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にレベルへのアクセスが行われた日付と時刻。
AW\$LOGICAL_NAME	NA
AW\$PARENT_NAME	<i>Dimdef</i> ディメンション。
AW\$ROLE	LEVELLIST
AW\$STATE	CREATED

Member_Levelrel リレーション

member_levelrel リレーションは、各ディメンション・メンバーのレベルを識別します。このリレーションを使用すると、レベル別のディメンション・メンバーの選択が容易になります。情報は、リレーショナル・ファクト表から取得されます。通常、このテキスト・ディメンションは *dimdef_LEVELREL* という名前を持ちます。

Member_Levelrel リレーションの内容

例 8-7 に、GLOBAL の CUSTOMER の *member_levelrel* リレーションを示します。

例 8-7 GLOBAL の CUSTOMER の Member_Levelrel リレーション

```
LIMIT customer TO '62' "Select customer 62
LIMIT customer ADD ANCESTORS USING customer_parentrel "Add ancestors
REPORT DOWN customer W 20 customer_levelrel
```

CUSTOMER	CUSTOMER_LEVELREL
62	SHIP_TO
21	WAREHOUSE
27	ACCOUNT
10	REGION
6	MARKET_SEGMENT
1	ALL_CUSTOMERS
7	TOTAL_MARKET

Member_Levelrel リレーションのプロパティ

表 8-7 に、*member_levelrel* リレーションの OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-7 Member_Levelrel リレーションのプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION
AW\$CREATEDBY	レベルの作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、レベルが AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にレベルへのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	Dimdef デイメンション。
AW\$ROLE	MEMBER_LEVELREL
AW\$STATE	CREATED

Member_Familyrel リレーション

Member_familyrel リレーションを使用すると、Oracle Discoverer のビューのパフォーマンスが向上します。これによって、クロス集計で単一行内の各ディメンション・メンバーの完全な親子関係が提供されます。ファミリー・リレーションの標準名は、*dimdef_FAMILYREL* です。

Member_Familyrel リレーションの内容

例 8-8 に、GLOBAL の CUSTOMER の Member_Familyrel リレーションを示します。

例 8-8 GLOBAL の CUSTOMER の Member_Familyrel リレーション

LIMIT customer TO '78' "Select customer 78
 LIMIT customer ADD ANCESTORS USING customer_parentrel "Add the ancestors
 LIMIT customer_hierlist TO 'SHIPMENTS' "Select the SHIPMENTS hierarchy

REPORT customer_familyrel

CUSTOMER_HIERLIST: SHIPMENTS

-----CUSTOMER_FAMILYREL-----							
-----CUSTOMER-----							
CUSTOMER_LEVEL							
LIST	78	21	31	10	2	1	7
-----	-----	-----	-----	-----	-----	-----	-----
TOTAL_MARKET	NA	NA	NA	NA	NA	NA	NA
MARKET_SEGMENT	NA	NA	NA	NA	NA	NA	NA
ACCOUNT	NA	NA	NA	NA	NA	NA	NA
ALL_CUSTOMERS	1	1	NA	1	NA	1	NA
REGION	10	10	NA	10	NA	NA	NA
WAREHOUSE	21	21	NA	NA	NA	NA	NA
SHIP_TO	78	NA	NA	NA	NA	NA	NA

Member_Familyrel リレーションのプロパティ

表 8-8 に、*member_familyrel* リレーションの OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-8 Member_Familyrel リレーションのプロパティ

プロパティ	値
AW\$CLASS	FEATURES
AW\$CREATEDBY	リレーションの作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、リレーションが AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にリレーションへのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	<i>Dimdef</i> デイメンション。
AW\$ROLE	MEMBER_FAMILYREL
AW\$STATE	CREATED

レベルのスタンダード・フォーム・メタデータ

レベルのスタンダード・フォーム・メタデータは、次のオブジェクトに格納されます。

- ALL_LEVELS ディメンション
- ALL_DESCRIPTIONS 変数
- DIM_LEVELS 値セット

ALL_LEVELS ディメンション

ALL_LEVELS ディメンションは、次の形式ですべてのレベルの名前を格納します。

`workspace.dimension.level.LEVEL`

例: `GLOBAL_AW.TIME.Quarter.LEVEL`

ALL_LEVELS は、ALL_OBJECTS CONCAT ディメンションのベース・ディメンションです。ALL_OBJECTS は、ALL_DESCRIPTIONS および AW_NAMES をディメンション化します。ALL_DESCRIPTIONS はレベルの値を提供しますが、AW_NAMES は提供しません。

レベルの ALL_DESCRIPTIONS 変数

ALL_DESCRIPTIONS 変数は、レベルの簡単な説明、詳細な説明および複数説明を格納します。すべてのレベルはメタデータから取得した簡単な説明を持ちますが、詳細な説明と複数説明は持たない場合もあります。

DIM_LEVELS 値セット

DIM_LEVELS 値セットは、各ディメンションに定義されているレベルを識別します。

スタンダード・フォームの属性

属性は、通常テキスト・データ型で変数として定義します。属性はディメンション・メンバーに関する情報を提供し、通常、リレーショナル・ディメンション表から取得されます。属性は、*dimdef* ディメンション、*hierlist* ディメンションおよび ALL_LANGUAGES ディメンションによってディメンション化されます。

ディメンション・メンバーはロード時にソートされ、ディメンション・メンバーを行ごとにアナリティック・ワークスペースにフェッチしたときの元の順序が、*dimension_ORDER* という名前の属性によって識別されます。

表 8-9 に、属性の OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-9 属性のプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION

表 8-9 属性のプロパティ

プロパティ	値
AW\$CREATEDBY	属性の作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、属性が AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後に属性へのアクセスが行われた日付と時刻。
AW\$LOGICAL_NAME	OLAP カタログのソース属性名。
AW\$PARENT_NAME	<i>Dimdef</i> デイメンション。
AW\$ROLE	ATTRDEF
AW\$STATE	CREATED
AW\$TYPE	現在、Long Description、Short Description、Time Span、End Date および DEFAULT_ORDER は特別な属性タイプとして使用されています。
SOURCE_DATATYPE	ソース列の基本データ型 (VARCHAR2 または DATE など)。これは、AWCREATE プログラムで使用されます。
SOURCE_DIMNAME	OLAP カタログのソース・デイメンション名。これは、AWCREATE プログラムで使用されます。
SOURCE_NAME	OLAP カタログのソース属性名。これは、AWCREATE プログラムおよびアクティブ・カタログで使用されます。
SOURCE_OWNER	OLAP カタログのソース・メタデータの所有者。これは、AWCREATE プログラムおよびアクティブ・カタログで使用されます。

スタンダード・フォームの属性の詳細は、C-14 ページの「[実装クラスのオブジェクト](#)」および C-24 ページの「[カタログ・クラスのオブジェクト](#)」を参照してください。

ALL_LANGUAGES デイメンション

ALL_LANGUAGES デイメンションを使用すると、アナリティック・ワークスペースで複数の言語をサポートできます。このデイメンションは初期状態で 1 つのメンバーを持ちます。このメンバーは、データベース (アナリティック・ワークスペース) の地域と言語を識別します (たとえば、AMERICAN_AMERICA)。

表 8-10 に、ALL_LANGUAGES デイメンションの OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、[付録 C](#) を参照してください。

表 8-10 ALL_LANGUAGES ディメンションのプロパティ

プロパティ	値
AW\$CLASS	CATALOG
AW\$CREATEDBY	AW\$CREATE
AW\$LASTMODIFIED	日付と時刻
AW\$ROLE	ALL_LANGUAGES
AW\$STATE	CREATED

属性のスタンダード・フォーム・メタデータ

属性のスタンダード・フォーム・メタデータは、次のオブジェクトに格納されます。

- ALL_ATTRIBUTES ディメンション
- ALL_DESCRIPTIONS 変数
- AW_NAMES 変数
- DIM_ATTRIBUTES 値セット

ALL_ATTRIBUTES ディメンション

ALL_ATTRIBUTES ディメンションは、次の形式ですべての属性の名前を格納します。

`workspace.dimension.attribute.ATTRIBUTE`

例: `GLOBAL_AW.TIME.End_Date.ATTRIBUTE`

ALL_ATTRIBUTES は、ALL_OBJECTS CONCAT ディメンションのベース・ディメンションです。ALL_OBJECTS は、ALL_DESCRIPTIONS および AW_NAMES をディメンション化するため、これらのカタログは各属性のエントリを持ちます。

属性の ALL_DESCRIPTIONS 変数

ALL_DESCRIPTIONS 変数は、属性の簡単な説明、詳細な説明および複数説明を格納します。すべてのオブジェクトはメタデータから取得した簡単な説明を持ちますが、詳細な説明と複数説明は持たない場合もあります。

属性の AW_NAMES 変数

AW_NAMES メジャーは、各属性の名前を次の形式で提供します。

`schema.workspace!attribute`

例: `GLOBAL_AW.GLOBAL!TIME_END_DATE`

スタンダード・フォームのメジャー

各メジャーは、変数と式の 2 つのワークスペース・オブジェクトで定義します。

スタンダード・フォームのメジャーの詳細は、C-14 ページの「[実装クラスオブジェクト](#)」および C-39 ページの「[拡張クラスオブジェクト](#)」を参照してください。

メジャー変数

初期のメジャー変数には、ベース・レベルのデータのみが含まれます。通常、このデータはリレーショナル・ファクト表から取得されます。集計プランをデプロイする場合、事前に計算された集計レベルも変数に含まれます。

メジャー変数のデータ型は、初期作成時にデータをロードする前に再定義しないかぎり、DECIMAL です。メジャー変数の標準名は、*measuredef_VARIABLE* です。

[表 8-11](#) にプロパティを示します。オブジェクト型に依存しないプロパティの詳細は、[付録 C](#) を参照してください。

表 8-11 メジャー変数のプロパティ

プロパティ	値
AW\$CLASS	EXTENSION
AW\$CREATEDBY	キューブの作成者。リフレッシュ・ウィザードでは、AW\$CREATE の値が必要です。これは、オブジェクトが AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にキューブへのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	メジャーの名前。
AW\$ROLE	VARIABLE
AW\$SEGWIDTH_CMD	セグメント・サイズを定義する CHGDFN コマンド。
AW\$STATE	CREATED

Measuredef 式

measuredef 式は、aggmap に格納されている一連の集計規則を使用して集計データを計算します。この標準名はメジャーの名前です。[表 8-12](#) にプロパティを示します。オブジェクト型に依存しないプロパティの詳細は、[付録 C](#) を参照してください。

表 8-12 Measuredef 式のプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION
AW\$COMPSPEC	配置されている現行の aggmap の名前。
AW\$CREATEDBY	メジャーの作成者。一部のツールでは、AW\$CREATE の値が必要な場合があります。これは、メジャーが AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にメジャーへのアクセスが行われた日付と時刻。
AW\$LOGICAL_NAME	OLAP カタログのメジャー名。
AW\$PARENT_NAME	キューブ名。
AW\$ROLE	MEASUREDEF
AW\$STATE	CREATED
MEASCOLS	アナリティック・ワークスペースのカタログの名前 (cube.MSCL)。この名前によって、ファクト表内のソース列が識別されます。AWCREATE プログラムで使用されます。
SOURCE_CUBENAME	OLAP カタログのソース・キューブの名前。これは、AWCREATE プログラムで使用されます。
SOURCE_NAME	OLAP カタログのソース・メジャー。これは、AWCREATE プログラムおよびアクティブ・カタログで使用されます。
SOURCE_OWNER	OLAP カタログのソース・メタデータの所有者。これは、AWCREATE プログラムおよびアクティブ・カタログで使用されます。

メジャーのスタンダード・フォーム・メタデータ

メジャーのスタンダード・フォーム・メタデータは、次のオブジェクトに格納されます。

- ALL_MEASURES ディメンション
- ALL_DESCRIPTIONS 変数
- AW_NAMES 変数
- CUBE_MEASURES 値セット

ALL_MEASURES ディメンション

ALL_MEASURES ディメンションは、次の形式ですべてのメジャーの名前を格納します。

```
workspace.cube.measure.MEASURE
```

例: GLOBAL_AW.UNITS_CUBE.UNITS.MEASURE

ALL_MEASURES は、ALL_OBJECTS CONCAT ディメンションのベース・ディメンションです。ALL_OBJECTS は、ALL_DESCRIPTIONS および AW_NAMES をディメンション化するため、これらのカタログは各メジャーのエントリを持ちます。

メジャーの ALL_DESCRIPTIONS 変数

ALL_DESCRIPTIONS 変数は、メジャーの簡単な説明、詳細な説明および複数説明を格納します。すべてのオブジェクトはメタデータから取得した簡単な説明を持ちますが、詳細な説明と複数説明は持たない場合もあります。

メジャーの AW_NAMES 変数

AW_NAMES メジャーは、各メジャーの名前を次の形式で提供します。

```
schema.workspace!measure
```

例: GLOBAL_AW.GLOBAL!UNITS

CUBE_MEASURES 値セット

CUBE_MEASURES 値セットは、各キューブのメジャーを識別します。

スタンダード・フォームのキューブ

キューブは、キューブのディメンション（エッジ）の名前を表示するテキスト・ディメンションとして実装されます。キューブ内のメジャーすべてに、デフォルトの集計マップおよびコンポジット・ディメンションも定義されます。

スタンダード・フォームのキューブの詳細は、C-14 ページの「[実装クラスのオブジェクト](#)」を参照してください。

Cubedef ディメンション

cubedef ディメンションは、キューブ内のメジャーについて *dimdef* ディメンションの名前（TIME および PRODUCT など）を表示します。このディメンションの標準名は、論理キューブの名前になります（UNITS_CUBE など）。作成オプションで接頭辞を指定した場合、名前に接頭辞が付きます。

Cubedef ディメンションの内容

[例 8-9](#) に、GLOBAL の UNITS_CUBE の *cubedef* ディメンションを示します。

例 8-9 GLOBAL の Units キューブのディメンション

```
REPORT units_cube

UNITS_CUBE
-----
CHANNEL
CUSTOMER
PRODUCT
TIME
```

Cubedef ディメンションのプロパティ

表 8-13 に、*cubedef* ディメンションの OLAP DML プロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-13 Cubedef のプロパティ

プロパティ	値
AGGMAPLIST	このキューブに定義されているすべての <i>aggmap</i> の名前を示す、単一行または複数行のテキスト文字列。これは、 <i>AWCREATE</i> プログラムで使用されます。
AW\$CLASS	IMPLEMENTATION
AW\$CREATEDBY	キューブの作成者。リフレッシュ・ウィザードおよび集計プラン・ツールでは、 <i>AW\$CREATE</i> の値が必要です。これは、オブジェクトが <i>AWCREATE</i> プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にキューブへのアクセスが行われた日付と時刻。
AW\$LOADPRGS	リレーショナル・スキーマからアナリティック・ワークスペースにデータをフェッチするために使用した OLAP DML プログラム名。
AW\$LOGICAL_NAME	ソース・キューブの論理名（OLAP カタログに定義されているキューブなど）。
AW\$LOOPSPEC	このキューブの変数の定義に使用したワークスペースのコンボジット。
AW\$PARENT_NAME	NA
AW\$ROLE	CUBEDEF
DISPLAY_NAME	OLAP カタログのソースの表示名。
FORMDIMS	メジャー式のディメンションの順序付けされたリスト。これは、 <i>AWCREATE</i> プログラムで使用されます。

表 8-13 Cubedef のプロパティ

プロパティ	値
LOADNAME	キューブの移入に使用するロード・プログラムの名前。これは、AWCREATE プログラムで使用されます。
LOADTYPE	作成の一部としてデータがロードされる場合は LOAD_DATA、DML ロード・プログラムを作成したが実行していない場合は LOAD_PROGRAM。これらは、DBMS_AWM.CREATE_AWCUBELOAD_SPEC のキーワードであり、AWCREATE プログラムで使用されます。
SOURCE_NAME	OLAP カタログのソース・キューブ。これは、AWCREATE プログラムおよびアクティブ・カタログで使用されます。
SOURCE_OWNER	OLAP カタログのソース・メタデータの所有者。これは、AWCREATE プログラムおよびアクティブ・カタログで使用されます。
SYS_DIMS	メジャー変数のディメンションの順序付けされたリスト。通常は、Time、その他のすべてのディメンションのコンポジットの順になります。これは、AWCREATE プログラムで使用されます。
SYS_DIMSML	キューブのディメンションのアルファベット順のリスト。これは、AWCREATE プログラムで使用されます。

Comspec 集計マップ

各キューブには、すべてのディメンション間で実行時の集計を指定するデフォルトの **aggmap** が作成されます。初期状態では、この **aggmap** は、キューブに関連付けられたすべてのメジャーの式によって参照されます。Analytic Workspace Manager のウィザードを使用して集計プランを作成およびデプロイする場合は、新しい **aggmap** を作成し、指定したメジャーの式を変更します。

デフォルトの **aggmap** の標準名は、*cubedef_AGGMAP_AWCREATEDDEFAULT_1* です（たとえば、*UNITS_CUBE_AGGMAP_AWCREATEDDEFAULT_1*）。

表 8-14 に、*comspec aggmap* のプロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-14 Comspec Aggmap のプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION
AW\$COUNTVARCMD	aggmap で平均を計算する場合に使用される整変数の名前。それ以外の場合は NA。

表 8-14 Comspec Aggmap のプロパティ

プロパティ	値
AW\$CREATEDBY	aggmap の作成者。一部のツールでは、AW\$CREATE の値が必要な場合があります。これは、aggmap が AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後に aggmap へのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	ワークスペース・キューブ名。
AW\$ROLE	COMSPEC
AW\$STATE	CREATED
ISDFLTAGGMAP	デフォルトの集計マップの場合は YES、初期作成後に作成された aggmap の場合は NO。これは、データのリフレッシュ時に使用されます。

Loopspec コンポジット・ディメンション

アナリティック・ワークスペース作成ウィザードを使用すると、デフォルトのコンポジットを受け入れたり、コンポジットを独自に定義することもできます。

デフォルトのコンポジットは、*cubedef_COMPOSITE* という名前が付き、**Time** ディメンションを除くキューブのすべてのディメンションで構成されます。ディメンションは、メンバーが最も多いものから最も少ないものに順序付けられます。

カスタム・コンポジットは、割り当てた名前および特性を持ちます。

表 8-15 に、*loopspec* コンポジット・ディメンションのプロパティを示します。オブジェクト型に依存しないプロパティの詳細は、付録 C を参照してください。

表 8-15 Loopspec コンポジットのプロパティ

プロパティ	値
AW\$CLASS	IMPLEMENTATION
AW\$CREATEDBY	コンポジットの作成者。一部のツールでは、AW\$CREATE の値が必要な場合があります。これは、コンポジットが AWCREATE プログラムで作成されたことを示します。
AW\$LASTMODIFIED	アナリティック・ワークスペースのツールで最後にコンポジットへのアクセスが行われた日付と時刻。
AW\$PARENT_NAME	<i>Cubedef</i> ディメンション。
AW\$ROLE	LOOPSPEC

表 8-15 *Loopspec* コンポジットのプロパティ

プロパティ	値
AW\$STATE	CREATED

loopspec コンポジットの詳細は、C-14 ページの「[実装クラスオブジェクト](#)」を参照してください。

キューブのスタンダード・フォーム・メタデータ

キューブのスタンダード・フォーム・メタデータは、次のオブジェクトに格納されます。

- ALL_CUBES ディメンション
- ALL_DESCRIPTIONS 変数
- AW_NAMES 変数
- CUBE_MEASURES 値セット

ALL_CUBES ディメンション

ALL_CUBES ディメンションは、次の形式ですべてのキューブの名前を格納します。

`workspace.cube.CUBE`

例: `GLOBAL_AW.UNITS_CUBE.CUBE`

ALL_CUBES は、ALL_OBJECTS CONCAT ディメンションのベース・ディメンションです。ALL_OBJECTS は、ALL_DESCRIPTIONS および AW_NAMES をディメンション化するため、これらのカタログは各キューブのエントリを持ちます。

キューブの ALL_DESCRIPTIONS 変数

ALL_DESCRIPTIONS 変数は、キューブの簡単な説明、詳細な説明および複数説明を格納します。すべてのオブジェクトはメタデータから取得した簡単な説明を持ちますが、詳細な説明と複数説明は持たない場合もあります。

キューブの AW_NAMES 変数

AW_NAMES メジャーは、各キューブの名前を次の形式で提供します。

`schema.workspace!cube`

例: `GLOBAL_AW.GLOBAL!UNITS_CUBE`

CUBE_MEASURES 値セット

CUBE_MEASURES 値セットは、各キューブのメジャーを識別します。

スタンダード・フォームのカタログ

データベース・スタンダード・フォームでは、オブジェクトのカタログ・クラスが必要です。これらのオブジェクトは、論理モデルを実装するアナリティック・ワークスペースのオブジェクトに関する情報を保持します。表 8-16 に、このようなオブジェクトを示します。カタログの詳細は、C-24 ページの「[カタログ・クラスのオブジェクト](#)」を参照してください。

表 8-16 スタンダード・フォームのカタログ

カタログ	オブジェクト型	内容
ALL_ATTRIBUTES	ディメンション	各ワークスペース・ディメンションの各属性のフルネーム。形式は、 <i>schema.dimension.attribute.ATTRIBUTE</i> です。
ALL_CUBES	ディメンション	各ワークスペース・キューブ（値がキューブのディメンションであるディメンション）のフルネーム。形式は、 <i>schema.cube.CUBE</i> です。
ALL_DESCRIPTIONS	変数	論理オブジェクトの簡単な説明、詳細な説明および複数説明を格納します。
ALL_DIMENSIONS	ディメンション	各ワークスペース・データ・ディメンション（データ・キューブで使用されるディメンション）のフルネーム。形式は、 <i>schema.dimension.DIMENSION</i> です。
ALL_HIERARCHIES	ディメンション	各ワークスペース・ディメンションの各階層のフルネーム。形式は、 <i>schema.dimension.hierarchy.HIERARCHY</i> です。
ALL_LEVELS	ディメンション	各ワークスペース・ディメンションの各レベルのフルネーム。形式は、 <i>schema.dimension.level.LEVEL</i> です。
ALL_MEASURES	ディメンション	各ワークスペース・メジャー（完全に解決済のメジャーを戻す式）のフルネーム。形式は、 <i>schema.measure.MEASURE</i> です。
ALL_OBJECTS	CONCAT ディメンション	ALL_DIMENSIONS、ALL_CUBES、ALL_MEASURES、ALL_HIERARCHIES、ALL_LEVELS および ALL_ATTRIBUTES。
AW_NAMES	変数	ソース・メタデータで定義されている各論理オブジェクトを実装する、アナリティック・ワークスペース・オブジェクトの名前。
CUBE_MEASURES	値セット	各キューブに属しているメジャーのリスト。
DEFAULT_HIER	リレーション	各ディメンションのデフォルトの階層のフルネーム。

表 8-16 スタンダード・フォームのカタログ

カタログ	オブジェクト型	内容
DIM_ATTRIBUTES	値セット	各ディメンションに属している属性のリスト。
DIM_HIERARCHIES	値セット	各ディメンションに属している階層のリスト。
DIM_LEVELS	値セット	各ディメンションに属しているレベルのリスト。

OLAP API イネーブラのカタログ

作成プロセスでは、OLAP API および BI Beans のイネーブラおよびアクティブ・カタログをサポートするために、アナリティック・ワークスペース内に多くのオブジェクトが作成されます。また、イネーブラによっても複数のオブジェクトが作成されます。表 8-17 に、OLAP API で使用されるカタログを示します。

注意： OLAP API イネーブラのカタログは、将来のソフトウェア・リリースで変更または廃止される場合があります。

表 8-17 OLAP API イネーブラのカタログ

カタログ	オブジェクト型	内容
__SYS_HIERCJT	結合	ファクト・ビューが必要なディメンション階層の組合せ。これは、有効化の処理で一時的に使用するために作成されます。
__SYS_HIERCJT_BUILD	結合	ファクト・ビューが必要なディメンション階層の組合せ。これは、キューブのリフレッシュの処理で一時的に使用するために作成されます。
cube_SYS_ENABLE	変数	アナリティック・ワークスペース・キューブのイネーブラで生成されるオブジェクト型、表型およびビューの名前を示す、複数行のテキスト文字列。
dimension_SYS_ENABLE	変数	アナリティック・ワークスペース・ディメンションのイネーブラで生成されるオブジェクト型、表型およびビューの名前を示す、複数行のテキスト文字列。
OLAP_SYS_ADTDIM	ディメンション	ビューを生成するために OLAP_TABLE ファンクションで使用されるオブジェクト型の名前。
OLAP_SYS_ADTTREL	リレーション	各リレーショナル・ビューについて OLAP_TABLE ファンクションで使用されるオブジェクト型の名前。
OLAP_SYS_ADTTBLDIM	ディメンション	ビューを生成するために OLAP_TABLE ファンクションで使用される表型の名前。

表 8-17 OLAP API イネーブラのカタログ

カタログ	オブジェクト型	内容
OLAP_SYS_ADTTBLREL	リレーション	各リレーショナル・ビューについて OLAP_TABLE ファンクションで使用される表型の名前。
OLAP_SYS_CUBEADTNAME_VAR	変数	各ファクト・ビューを生成するために OLAP_TABLE ファンクションで使用されるオブジェクト型。
OLAP_SYS_CUBEAWOWNER_VAR	変数	各ファクト・ビューのスキーマの所有者。これは、ALL_OLAP2_CUBE_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_CUBEHIERCOMBO_VAR	変数	ファクト・ビューで示されるディメンション階層の各組合せの整数値。これは、ALL_OLAP2_CUBE_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_CUBEHIERCOMBOSTR_VAR	変数	各ファクト・ビューで示されるディメンションおよび階層を識別するテキスト文字列。これは、ALL_OLAP2_CUBE_ENABLED_VIEWS アクティブ・カタログで使用されます。
OLAP_SYS_CUBENAME_DIM	ディメンション	アナリティック・ワークスペースのキューブの名前。
OLAP_SYS_CUBENAME_REL	リレーション	各ファクト・ビューで示されるアナリティック・ワークスペース・キューブ。
OLAP_SYS_CUBENAME_VAR	変数	各ファクト・ビューで示されるアナリティック・ワークスペース・キューブ。これは、ALL_OLAP2_CUBE_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_CUBETBLNAME_VAR	変数	各ファクト・ビューを生成するために OLAP_TABLE ファンクションで使用される表型。
OLAP_SYS_CUBEUSERVIEW_VAR	変数	CWM2_OLAP_CUBE.SET_CUBE_NAME プロシージャを使用してワークスペース・キューブに割り当てる新しい名前。新しい名前が未定義の場合は NA。これは、ALL_OLAP2_CUBE_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_CUBEVALSET	値セット	キューブのリフレッシュ時のファクト・ビューの名前。名前が未定義の場合の場合は NA。
OLAP_SYS_CUBEVIEW_DIM	ディメンション	アナリティック・ワークスペースに定義されているファクト・ビューの名前。これは、ALL_OLAP2_CUBE_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_DIMADTNAME_VAR	変数	各ディメンション・ビューについて OLAP_TABLE ファンクションで使用されるオブジェクト型の名前。

表 8-17 OLAP API イネーブラのカタログ

カタログ	オブジェクト型	内容
OLAP_SYS_DIMAWOWNER_VAR	変数	各ディメンション・ビューのスキーマの所有者。これは、ALL_OLAP2_DIM_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_DIMHIERNAME_VAR	変数	各ディメンション・ビューで示される階層の名前。これは、ALL_OLAP2_DIM_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_DIMHIERPOS_VAR	変数	<i>dimension_HIERLIST</i> 階層ディメンションにおける各階層の位置 (数値)。
OLAP_SYS_DIMNAME_DIM	ディメンション	アナリティック・ワークスペースのディメンションの名前。
OLAP_SYS_DIMNAME_REL	リレーション	各ディメンション・ビューで示されるアナリティック・ワークスペース・ディメンション。
OLAP_SYS_DIMNAME_VAR	変数	各ディメンション・ビューで示されるディメンションのアナリティック・ワークスペース名。これは、ALL_OLAP2_DIM_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_DIMTBLNAME_VAR	変数	各ディメンション・ビューについて OLAP_TABLE ファンクションで使用される表型の名前。
OLAP_SYS_DIMUSERVIEW_VAR	変数	CWM2_OLAP_DIMENSION.SET_DIMENSION_NAME プロシージャを使用してワークスペース・キューブに割り当てる新しい名前。新しい名前が未定義の場合は NA。これは、ALL_OLAP2_DIM_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_DIMVALSET	値セット	ディメンションのリフレッシュ時のディメンション・ビューの名前。名前が未定義の場合の場合は NA。
OLAP_SYS_DIMVIEW_DIM	ディメンション	各階層のリレーショナル・ディメンション・ビューの名前。これは、ALL_OLAP2_DIM_ENABLED_VIEW アクティブ・カタログで使用されます。
OLAP_SYS_LIMITMAP	変数	各リレーショナル・ビューについて OLAP_TABLE ファンクションで使用される制限マップ。
OLAP_SYS_VIEWDIM	ディメンション	アナリティック・ワークスペースに定義されているリレーショナル・ビューの名前。

AWCREATE カタログ

作成プロセスおよびリフレッシュ・プロセスで使用され、アナリティック・ワークスペースに現存しているカタログがいくつかあります。カタログの中には、Oracle Discoverer の有効

化プロセスで一時的に使用されるものもあります。表 8-18 に、AWCREATE カタログを示します。

注意： AWCREATE カタログは、将来のソフトウェア・リリースで変更または廃止される場合があります。

表 8-18 AWCREATE カタログ

カタログ	オブジェクト型	内容
<i>cube_HIERCJT</i>	結合	キューブのディメンションの階層の名前。
<i>cube_HIERCJT.DMKY</i>	変数	キューブのディメンションに対するソース・ディメンション表のキー列の名前。
<i>cube_HIERCJT.DMLV</i>	変数	データが格納されるディメンション・レベルの名前。
<i>cube_HIERCJT.FT</i>	変数	キューブのソースであるファクト表の名前。
<i>cube_HIERCJT.HC</i>	変数	整数値。
<i>cube_measure.MSCL</i>	変数	メジャーのソース列の名前。
<i>dimension_attribute_SRCATTRCOL</i>	変数	階層別およびレベル別の属性のソース列の名前。
<i>dimension_attribute_SRCATTROWNER</i>	変数	階層別およびレベル別の属性のソース・ディメンション表のスキーマの所有者名。
<i>dimension_attribute_SRCATTRTBL</i>	変数	階層別およびレベル別の属性のソース・ディメンション表の名前。
<i>dimension_LEVELCOLLIST</i>	ディメンション	整数値。
<i>dimension_LEVELCOLMAP</i>	変数	アナリティック・ワークスペースの各ディメンション・メンバーに対応するソース・ディメンション値。値は、作成時に接頭辞を取得できます。
<i>dimension_SRCCOMPOSITE</i>	コンポジット	<i>dimension_HIERLIST</i> ディメンション、 <i>dimension_LEVELLIST</i> ディメンション、 <i>dimension_LEVELCOLLIST</i> ディメンションで構成される コンポジット・ディメンション。
<i>dimension_SRCLVLCOL</i>	変数	レベルのソース列の名前。
<i>dimension_SRCLVLOWNER</i>	変数	ソース・ディメンション表のスキーマの所有者の名前。
<i>dimension_SRCLVLPNTCOL</i>	変数	親レベルのソース列の名前。
<i>dimension_SRCLVLTBL</i>	変数	ソース・ディメンション表の名前。

第 III 部

他のソースからのデータ取得

第 III 部では、スター・スキーマまたはスノーフレーク・スキーマ以外のソースのデータを使用してアナリティック・ワークスペースを新規作成する方法、または既存のアナリティック・ワークスペースを整備する方法について説明します。使用できるデータは、アナリティック・ワークスペースで利用できる分析機能によって生成されたデータ、またはフラット・ファイルなどの外部ソースからのデータです。

ここでは、次の項目について説明します。

- [第 9 章「スタンダード・フォームのアナリティック・ワークスペースへのメジャーの追加」](#)
- [第 10 章「将来の業績の予測」](#)
- [第 11 章「他のソースからのデータの取得」](#)

スタンダード・フォームのアナリティック・ワークスペースへのメジャーの追加

この章では、スタンダード・フォームのアナリティック・ワークスペースに永続的に追加するメジャーを新規作成する方法について説明します。この章で説明する方法を使用すると、（オンデマンドで計算するのではなく）データを格納することが可能で、なおかつアナリティック・ワークスペース内の他のメジャーと同じように使用可能なカスタム・メジャーを定義できます。ただし、このプロセスは、[第7章](#)で説明した DBMS_AW_UTILITIES や OLAP_EXPRESSION を使用するプロセスよりも複雑です。

高度な計算手法（予測やアロケーションなど）を使用したり、外部ソースからデータをロードしたりして、新しいワークスペース・オブジェクトを移入することもできます。

OLAP DML コマンドの様々な実行方法についても説明します。

この章では、次の項目について説明します。

- [スタンダード・フォームのアナリティック・ワークスペースでの操作](#)
- [OLAP DML コマンドの実行方法](#)
- [キューブへのカスタム・メジャーの追加](#)
- [事例: Global アナリティック・ワークスペースへのメジャーの追加](#)

スタンダード・フォームのアナリティック・ワークスペースでの操作

Analytic Workspace Manager および現行バージョンのツールは、データベース・スタンダード・フォームのアナリティック・ワークスペースでのみ使用できます。[第8章](#)で説明されているように、データベース・スタンダード・フォーム（単にスタンダード・フォームとも呼ぶ）は、存在している必要のあるオブジェクトのタイプ、それらに割り当てられている必要のある OLAP DML プロパティ、およびワークスペース・オブジェクトを登録するためのアナリティック・ワークスペース内の各種カタログを識別します。

スタンダード・フォームに準拠することにより、各ツールは、データの集計やリフレッシュ、ビューおよびメタデータの生成といったジョブを実行できるようになります。スタンダード・フォームに準拠していない場合、各ツールは、論理多次元モデル内のワークスペース・オブジェクトの機能を識別できません。

アナリティック・ワークスペースが存続している間、新しいデータソースからメジャーを追加したり、予測やアロケーションなどの高度な計算手法を使用して永続的なカスタム・メジャーを定義したりすることが必要になる場合があります。

これらのメジャーを手動でリフレッシュする必要がある場合には、集計ツールや有効化ツールを引き続き新しいメジャーに対して使用できると便利です。

これらのツールで新しいメジャーにアクセスできるようにするには、次の手順を実行します。

1. 適切なワークスペース・オブジェクトを定義します（式またはメジャー、あるいはその両方）。
2. そのオブジェクトに、適切な値の OLAP DML プロパティを設定します。
3. そのオブジェクトをスタンダード・フォームのワークスペース・カタログに登録します。

この章では、これらの手順を実行する方法について説明します。

参照：

- カスタム・メジャーをアナリティック・ワークスペースに追加する別の方法については、[第 7 章](#)を参照してください。
- スタンダード・フォームのアナリティック・ワークスペース内のカタログ、オブジェクトおよびプロパティについては、[第 8 章](#)を参照してください。
- フラット・ファイルなどの外部ソースから新しいメジャーを移入する方法については、[第 11 章](#)を参照してください。

OLAP DML コマンドの実行方法

アナリティック・ワークスペースを操作する際、次のいずれかの方法で OLAP DML コマンドを OLAP エンジンに発行して、コマンドを実行できます。

- **Analytic Workspace Manager** のダイアログ・ボックスを使用して、アナリティック・ワークスペースを作成し、ディメンション、変数、モデル、**aggmap** などのワークスペース・オブジェクトを定義および変更する。
- **OLAP Worksheet** で、OLAP DML コマンドを発行するための対話的 OLAP セッションを開く。OLAP Worksheet は、**Analytic Workspace Manager** から実行できます。**Analytic Workspace Manager** でアタッチしたワークスペースは、ダイアログ・ボックス

または OLAP Worksheet のどちらからでも変更を加えることができます。これらは同じセッションを共有します。

- (SQL*Plus などの) SQL セッションで、OLAP DML コマンドを DBMS_AW.EXECUTE PL/SQL プロシージャへのコールに埋め込む。

この章では、できるだけ多くの Analytic Workspace Manager のダイアログ・ボックスの使用方法を示します。OLAP DML に習熟してきたら、他の方法の方が便利に感じられるかもしれません。

SQL ベースまたは Java ベースのアプリケーションを開発する場合は、次のようにして OLAP DML コマンドを埋め込むこともできます。

- SQL プログラムでは、DBMS_AW パッケージのプロシージャを使用して、OLAP DML コマンドを埋め込むことができます。
- Java プログラムでは、OLAP API の SPLExecutor クラスを使用して、OLAP DML コマンドを埋め込むことができます。

DBMS_AW パッケージと OLAP Worksheet のどちらも、1 つの作業環境内に SQL コマンドと OLAP DML コマンドを混在させることができます。

参照：

- SPLExecutor クラスの詳細は、OLAP API Javadoc を参照してください。
- DBMS_AW パッケージのプロシージャの詳細は、『Oracle OLAP リファレンス』を参照してください。

Analytic Workspace Manager を使用した OLAP DML の実行

第 6 章「アナリティック・ワークスペースの作成」では、Analytic Workspace Manager の各種ウィザードを使用してアナリティック・ワークスペースを作成および管理する方法について説明しました。これらのウィザードは、OLAP カタログ・ビューから使用できます。

Analytic Workspace Manager のオブジェクト・ビューには、アナリティック・ワークスペースで使用可能なすべてのオブジェクト型を定義するためのプロパティ・シートおよびメニューがあります。ウィザード、プロパティ・シートおよびメニューでの選択により、OLAP DML コマンドが OLAP エンジンに送信され、実行されます。

たとえばディメンションを作成するには、「Create Dimension」ダイアログ・ボックスを開き、プロパティ・シートを使用してディメンションを定義します。「Create」ボタンをクリックすると、アナリティック・ワークスペースで OLAP DML の DEFINE DIMENSION コマンドが作成および実行されます。

この方法は、特にウィザードで生成されたアナリティック・ワークスペースを拡張する場合に便利です。式を作成したり、aggmap に変更を加えたりできます。オブジェクト・ビューでオブジェクトを選択すると、プロパティ・ページでその特性の多くを変更できます。ただ

し、ディメンションや変数のデータ型など、オブジェクトの作成後には変更できない特性もあります。このような特性はグレー表示されます。

新しいオブジェクトを移入する場合は、OLAP Worksheet で適切な DML コマンドを実行します。Analytic Workspace Manager から OLAP Worksheet を実行すると、それまでと同じセッションに接続されます。OLAP Worksheet で行ったすべての変更は、ただちに Analytic Workspace Manager に反映されます。その逆の場合も同様です。これにより、グラフィカルなインタフェースとコマンドラインのインタフェースを同一セッション内で使い分けることができます。

OLAP Worksheet を使用した OLAP DML の実行

OLAP Worksheet は、OLAP DML に習熟しているユーザー、またはアナリティック・ワークスペースで複雑な開発作業を行っているユーザーに最適な環境です。OLAP セッションを開き、OLAP DML の豊富な機能を最大限に活用しながら対話的に作業できます。OLAP Worksheet には、プログラム、モデルおよび **aggmap** を記述するためのエディタが用意されています。また、SQL モードに切り替えて、リレーショナル表およびビューに SQL コマンドを発行することもできます。

OLAP Worksheet は、Analytic Workspace Manager から開くことができます。

手順 : Analytic Workspace Manager から OLAP Worksheet を開く方法

1. Analytic Workspace Manager を開きます。
2. データベースに接続します。
3. 「Tools」メニューから「**OLAP Worksheet**」を選択します。

OLAP Worksheet のウィンドウが開きます。Analytic Workspace Manager でアナリティック・ワークスペースをアタッチしている場合は、そのワークスペースが OLAP Worksheet のセッションにアタッチされます。

4. OLAP DML コマンドを実行するには、ウィンドウ最下部の入力ペインにコマンドを入力します。

たとえば、アタッチされているアナリティック・ワークスペースのリストを表示するには、次のコマンドを発行します。

```
AW LIST
```

EXPRESS ワークスペースは常にアタッチされている必要があることに注意してください。

注意： OLAP Worksheet と Analytic Workspace Manager は同じセッションを共有しているので、この2つのアプリケーションを切り替える際には注意が必要です。オブジェクト・ビューでのアクションは、OLAP Worksheet で発行するコマンドに影響する場合があります。LISTNAMES や DEFINE などの一部のコマンドは最初のワークスペースに対してのみ機能するので、AW LIST コマンドを使用してアナリティック・ワークスペースのアタッチ順序を確認してください。また、NAME ディメンションのステータスを使用する EXPORT などのコマンドを発行する前には、LIMIT NAME TO ALL コマンドを発行してください。

手順：OLAP Worksheet のエディタの使用

「Edit」ウィンドウを使用すると、プログラム、モデルまたは aggmap の内容を変更できます。また、Analytic Workspace Manager のプロパティ・ページを使用して、これらのオブジェクトを編集することもできます。ただし、このページでオブジェクトを実行することはできません。

ディメンション、変数、リレーション、値セットまたはその他のデータ・コンテナの内容をエディタで変更することはできません。

1. プログラム・オブジェクトの内容を追加するには、次のコマンドを発行して「Edit」ウィンドウを開きます。

```
EDIT program_name
```

例: EDIT CREATE_MEASURE

PROGRAM がデフォルトのオブジェクト型です。必要に応じて別の型を指定する必要があります。たとえば、集計マップを編集する場合は、EDIT AGGMAP units_cube_aggmap のようなコマンドを発行します。

2. プログラムに追加する OLAP コマンドを入力します。
3. プログラムの編集が終了したら、エディタの「File」メニューから「Save」を選択してから、「Close」を選択します。
4. プログラムをコンパイルおよび実行するには、次のコマンドを発行します。

```
CALL program_name
```

注意： プログラムは自動的にコンパイルされるので、COMPILE コマンドはオプションです。ただし、COMPILE コマンドを個別に実行すると、OLAP DML コマンドの構文エラーをすぐに確認できるので便利です。

5. SQL コマンドを発行するには、「Options」メニューから「SQL Mode」を選択します。OLAP DML コマンドの発行を再開するには、「SQL Mode」を解除します。

例 9-1 にサンプル・セッションを示します。このセッションでは、CREATE_MEASURE というプログラムを OLAP Worksheet で作成、コンパイルおよび実行します。

例 9-1 OLAP Worksheet での OLAP DML プログラムの作成

```
DEFINE create_measure PROGRAM

LD Define a database standard form measure

EDIT create_measure
.
.          " Enter program code
.          " Choose Save

CALL create_measure
```

DBMS_AW.EXECUTE を使用した OLAP DML の実行

DBMS_AW.EXECUTE プロシージャを使用すると、SQL セッション中にいつでも OLAP DML コマンドを発行できます。OLAP DML コマンドの出力を表示するには、セッション中に次の SQL コマンドを一度発行します。

```
SET SERVEROUT ON FORMAT WRAPPED
```

DBMS_AW.EXECUTE コマンドの書式

DBMS_AW.EXECUTE の基本的な書式を次に示します。一重引用符で囲んだ中に、セミコロンで区切った 1 つ以上の OLAP DML コマンドを指定できます。

```
EXECUTE DBMS_AW.EXECUTE('dml_command_1; dml_command_2; dml_command_n');
```

次の例の最初のコマンドは、GLOBAL アナリティック・ワークスペースを開きます。2 つ目のコマンドは、SALES_PP というメジャーを指定して、このメジャーに方程式を割り当てます。

```
EXECUTE DBMS_AW.EXECUTE('AW ATTACH global RW');
EXECUTE DBMS_AW.EXECUTE('-
    CONSIDER sales_pp; EQ LAG(sales, 1, time, LEVELREL time_levelrel)');
```

SQL からの DML プログラムへの内容の追加

プログラムの内容は、編集が容易なテキスト・ファイルに定義できます。次の手順を実行します。

1. データベース・ディレクトリ・オブジェクトを定義します（未定義の場合）。

```
CREATE DIRECTORY directory AS 'path_name';
GRANT permission ON DIRECTORY directory TO users;
```


2. アナリティック・ワークスペースを開きます（開いていない場合）。

```
EXECUTE DBMS_AW.EXECUTE('AW CREATE aw_name ');
```

または

```
EXECUTE DBMS_AW.EXECUTE('AW ATTACH aw_name RW');
```

OLAP DML プログラムは、データが格納されているアナリティック・ワークスペースとは別のワークスペースで開発することをお勧めします。

3. プログラム・オブジェクトを作成し、オプションとして説明を指定します。次の構文は新しいプログラムを定義します。

```
EXECUTE DBMS_AW.EXECUTE('DEFINE object PROGRAM; LD object description');
```

4. テキスト・エディタを開き、次の内容のファイルを作成します。

```
CONSIDER program_name
PROGRAM
.
.
" OLAP DML commands
.
END
```

ヒント： 複数のウィンドウを開くことのできるオペレーティング・システムを使用している場合は、1つのウィンドウをSQLセッション用に、もう1つをテキスト・ファイルの編集用に開くことができます。

5. 作成したテキスト・ファイルを実行します。

```
EXECUTE DBMS_AW.EXECUTE('INFILE directory/filename');
```

6. プログラムをコンパイルおよび実行します。

```
EXECUTE DBMS_AW.EXECUTE('CALL program_name');
```

例 9-2 にサンプル・セッションを示します。このセッションでは、CREATE_MEASURE というプログラムをSQLセッションで作成、コンパイルおよび実行します。

例 9-2 SQL*Plus での OLAP DML プログラムの作成

```
% sqlplus
.
.
.
SQL> CREATE DIRECTORY olapfiles AS '/users/oracle/olapfiles';
SQL> GRANT all ON DIRECTORY olapfiles TO ALL;
SQL> EXECUTE DBMS_AW.EXECUTE('AW ATTACH global_programs RW');
SQL> EXECUTE DBMS_AW.EXECUTE('DEFINE create_measure PROGRAM; LD Get measures');
```

```
-- Create a file named getmeas.inf in directory olapfiles with the
-- contents of the program. Start with the template and edit it for
-- the sample data.
```

```
SQL> EXECUTE DBMS_OLAP.EXECUTE('INFILE olapfiles/getmeas.inf');
```

```
SQL> EXECUTE DBMS_OLAP.EXECUTE('CALL create_measure');
```

キューブへのカスタム・メジャーの追加

アナリティック・ワークスペース内の変数のほとんどは、ソースのスター・スキーマまたはスノーフレーク・スキーマ内のベース・レベルのデータから作成されます。しかし、分析の結果を変数に格納することが必要な場合もあります。たとえば、予測を生成する場合、その予測を格納するターゲット変数を識別する必要があります。

第8章で説明したように、スタンダード・フォームのワークスペースでは、論理メジャーは変数および式で実装されます。データベース・スタンダード・フォームのカスタム・メジャーを追加するには、これらのオブジェクトを手動で作成する必要があります。その後、イネーブラがカスタム・メジャーを他のメジャーとまったく同じようにビューに格納します。アプリケーション側からは、通常のメジャーもカスタム・メジャーも同じように見えます。

スタンダード・フォームのメジャー変数の定義

アナリティック・ワークスペース内の既存のメジャーから、カスタム・メジャーをその場で計算するための式を定義するだけであれば、この手順は省略してかまいません。同じメジャー変数から、必要なだけカスタム・メジャーを定義できます。

ただし、予測やその他の分析の結果を格納したり、別のソースからデータをロードしたりする場合は、この手順を実行して **Analytic Workspace Manager** でメジャー変数を定義する必要があります。

ヒント： キューブ内の変数と同様の変数（データ型を含む）を定義する場合は、オブジェクト・ビューでその変数を右クリックし、「**Create Like**」を選択します。あとは、新しい変数のプロパティに変更を加えるだけです（表 9-1 を参照）。この方法の例については、9-17 ページの「**GLOBAL への新規変数の作成**」を参照してください。

1. オブジェクト・ビューを開き、目的のアナリティック・ワークスペースのフォルダを開きます。
2. 「Variable」フォルダを右クリックし、「**Create Variable**」を選択します。
「Create Variable」ダイアログ・ボックスが表示されます。

3. 「Basic」 ページで、変数の名前、説明およびデータ型を指定します。ワークスペース内の他の変数の名前と揃えるため、名前の末尾には必ず「_VARIABLE」を付け、先頭には任意でキューブ名（「SALES_CUBE_」など）を付けます。「Help」をクリックすると、各項目の詳細を参照できます。
4. 「Dimensions」 ページで変数のディメンションを選択し、それらを適切な順序でリストします。

注意： パフォーマンスを向上させるためには、適切な順序でリストすることが非常に重要です。

このメジャーを既存のキューブに追加する場合は、そのキューブの他の変数と同じようにこのメジャーをディメンション化します。ほとんどの場合、Time ディメンションを先頭に、その後に残りすべてのディメンションのコンポジットをリストします。それ以外の場合は、「Help」をクリックしてディメンションの順序についての情報を参照してください。

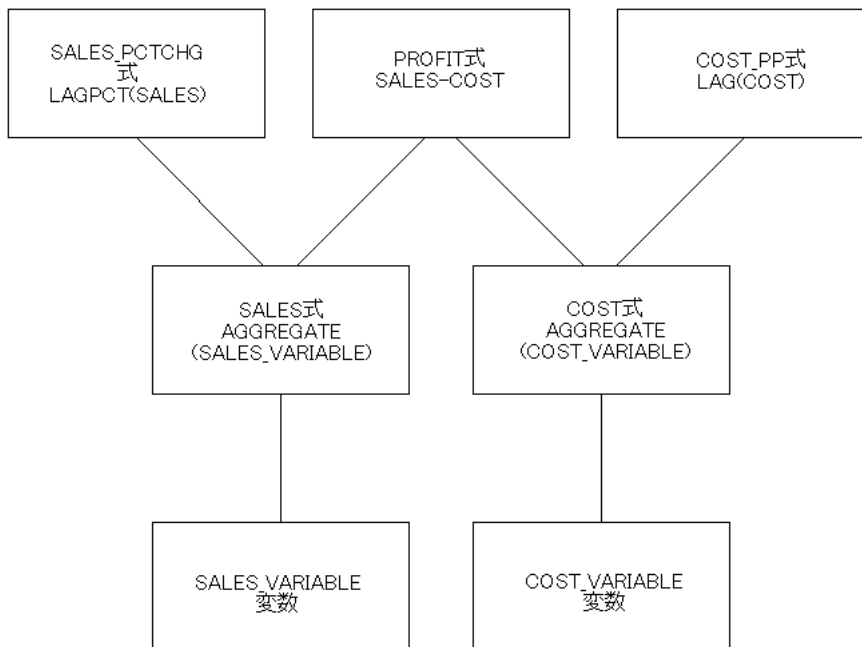
5. 「Properties」 ページで、8-19 ページの表 8-11 に示したプロパティを定義します。このメジャーを既存のキューブに追加する場合は、他の変数のプロパティ設定の多くと同じ設定にできます。それ以外の場合は、「Help」内でセグメント・サイズについての情報を検索してください。
6. 「File」 メニューから「Save」を選択して、アナリティック・ワークスペースおよび現在のスキーマ内のすべてのオブジェクトを更新します。
7. 変数を移入します。このためには、アナリティック・ワークスペース内の既存のメジャーに対して計算を実行するか、または外部データソースから移入します（第 11 章を参照）。

式の定義

1 つの変数は、多数の式のデータのソースにすることができます。アプリケーションは、変数ではなく式（または式のリレーショナル・ビュー）に対して問合せを発行します。したがって、式の名前がメジャーの名前になります。

各データ変数には、対応する式があります。この式は、方程式の中で AGGREGATE ファンクションを使用して、実行時にデータを集計します。データを操作する追加の式を作成できるので、OLAP DML に豊富に用意されているファンクションおよび演算子を利用することにより、情報量の多いデータをアナリティック・ワークスペースに追加できます。これらのワークスペース・オブジェクトの関係を図 9-1 に示します。

図 9-1 式と変数の関係



注意： ソース変数にすでに式が定義されている場合に、別の式を簡単に追加するには、オブジェクト・ビューで既存の式を右クリックし、「**Create Like**」を選択します。そして、式を置き換え、新しい式のプロパティに変更を加えます。この方法の例については、9-19 ページの「[メジャー式の作成](#)」を参照してください。

式を定義するには、次の手順を実行します。

1. オブジェクト・ビューを開き、目的のアナリティック・ワークスペースのフォルダを開きます。
2. 「Formulas」フォルダを右クリックし、「**Create Formula**」を選択します。
「Create Formula」ダイアログ・ボックスが表示されます。
3. 「Basic」ページで、式の名前、説明およびデータ型を指定します。式には、ソース変数と同じデータ型を選択します。「**Help**」をクリックすると、各項目の詳細を参照できます。

4. 「Dimensions」 ページで式のディメンションを選択し、それらを適切な順序でリストします。

ソース変数のベース・ディメンションをリストします。コンポジットを指定するのではなく、コンポジットを構成するディメンションをリストされている順序で指定します。パフォーマンスを向上させるには、ソース変数のディメンション順序を式に反映することが重要です。

たとえば、ソース変数が TIME および UNITS_CUBE_COMPOSITE でディメンション化されている場合、式を TIME、CUSTOMER、PRODUCT および CHANNEL で（この順序で）ディメンション化します。これは、UNITS_CUBE_COMPOSITE が CUSTOMER、PRODUCT および CHANNEL でディメンション化されているためです。

「Dimensions」 フォルダからコンポジットを選択し、その「Dimensions」 プロパティ・シートを表示すると、コンポジットのディメンションを確認できます。

5. 「Expression」 ページで、式の数式を入力します。

集計を行うためには、次の基本構文を使用して AGGREGATE ファンクションをコールします。

```
AGGREGATE(variable USING aggmap)
```

6. 「Properties」 ページで、8-20 ページの表 8-12 に示したプロパティを定義します。

7. SOURCE_CUBENAME、SOURCE_NAME および SOURCE_OWNER の各プロパティを削除します。

これらのプロパティはメジャーのリレーショナル・データソースを識別するもので、式の定義では必要ありません。

新規メジャーの登録

Analytic Workspace Manager のイネーブラやリフレッシュ・ウィザードなどのツールは、ワークスペースに格納されたメタデータを使用してオブジェクトを識別します。メタデータは、ディメンション、変数および値セットとして実装されるスタンダード・フォームのカタログに格納されます。新しいメジャーを作成したら、それをいくつかのカタログに登録する必要があります。登録では、データをワークスペース・オブジェクトに追加することになるので、OLAP Worksheet および OLAP DML を使用してメジャーを登録する必要があります。

新しいメジャーの登録には、4つのカタログが関係します。そのプロパティ・シートは、Analytic Workspace Manager で確認できます。または、OLAP Worksheet で次のコマンドを発行して、その定義を表示することもできます。

```
DESCRIBE all_measures all_descriptions aw_names cube_measures
```

ALL_MEASURES ディメンション

ALL_MEASURES **ディメンション**は、アナリティック・ワークスペース内のすべてのメジャーのリストです。REPORT コマンドを発行すると、ディメンションやメジャーなどのデータ・コンテナの内容を表示できます。基本構文は次のとおりです。

```
REPORT object
```

ALL_MEASURES ディメンションの内容を表示するには、次の OLAP DML コマンドを発行します。

```
REPORT W 60 all_measures
```

W パラメータは、レポートの列幅を指定します。

ディメンション・メンバーの追加

MAINTAIN コマンドを使用すると、ディメンション・メンバーの追加、削除および並べ替えを行うことができます。基本構文は次のとおりです。

```
MAINTAIN dimension ADD member
```

メジャーの名前は、次のような詳細な形式になります。

```
schema.cube.formula.MEASURE
```

メジャーの名前を ALL_MEASURES ディメンションに追加するには、次のコマンド構文を使用します。

```
MAINTAIN ALL_MEASURES ADD 'detailed_measure_name'
```

次に例を示します。

```
MAINTAIN ALL_MEASURES ADD 'GLOBAL.UNITS_CUBE.PROFIT.MEASURE'
```

注意： OLAP DML は、コマンドおよびワークスペース・オブジェクトの名前については、大 / 小文字を区別しません。したがって、MAINTAIN ALL_MEASURES、maintain all_measures、mAiNtAiN all_MEASures は同一のものと解釈されます。ただし、テキスト文字列（ディメンション・メンバーを含む）については、大 / 小文字が区別されます。したがって、'global.units_cube.profit.measure' と 'Global.Units_Cube.Profit.Measure' は別の値として扱われます。テキスト文字列は、必ず一重引用符で囲みます。

アナリティック・ワークスペースへの変更の保存

再度 REPORT コマンドを発行して変更が正しく行われていることを確認してから、次のコマンドを発行して変更内容を保存します。

```
UPDATE; COMMIT
```

UPDATE コマンドは、アナリティック・ワークスペースが格納されている LOB 表に変更内容をコピーします。COMMIT コマンドは SQL の COMMIT を発行し、そのセッションにおけるデータベースへの変更をすべて保存します。アナリティック・ワークスペースへの変更を保存し、将来のセッションで利用できるようにするには、両方のコマンドをこの順序で発行する必要があります。

ALL_DESCRIPTIONS 変数

ALL_DESCRIPTIONS 変数は、各ディメンション・メンバーの簡単な説明、詳細な説明および複数説明を格納します。これらの説明は、表示用に使用できます。ALL_DESCRIPTIONS は、ALL_OBJECTS、ALL_DESCTYPES および ALL_LANGUAGES でディメンション化されています。

- ALL_OBJECTS は CONCAT ディメンションです。つまり、ディメンション・メンバーの連結リストにある 2 つ以上のディメンションで構成されているということです。ALL_OBJECTS は、ALL_DIMENSIONS、ALL_CUBES、ALL_MEASURES、ALL_HIERARCHIES、ALL_LEVELS および ALL_ATTRIBUTES で構成されています。ディメンション・メンバーのメンテナンスおよび選択は、CONCAT ディメンションである ALL_OBJECTS に対して直接行われるのではなく、これらのベース・ディメンションに対して行われます。
- ALL_DESCTYPES は、そのメンバーとして LONG、SHORT および PLURAL を示します。ALL_DESCTYPES でディメンション化されたオブジェクトは、これらの型の複数の記述子を持つことができます。
- ALL_LANGUAGES は、アナリティック・ワークスペースでサポートされている言語を示します。初期状態では、AMERICAN_AMERICA などのデータベース言語が含まれます。ALL_LANGUAGES に言語を追加すると、これによってディメンション化されているオブジェクトは、テキストを複数の言語で表示できるようになります。

有効なディメンション・メンバー数の制限

アナリティック・ワークスペースでは、オブジェクトのデータは初期状態ですべて選択されています。つまり、**ステータス**が設定されています。データのサブセットを表示したり操作したりするには、LIMIT コマンドを使用して有効な値の数を制限する必要があります。LIMIT コマンドは、SQL の SELECT 文の WHERE 句と似ています。ただし、アナリティック・ワークスペースでは、このコマンドで選択した内容は明示的に変更するまで後続のコマンドでも有効です。

LIMIT は、ディメンションに対して機能し、ディメンション・メンバーを選択するためのオブションが数多く用意されています。LIMIT コマンドの最も基本的な書式は次のとおりです。

```
LIMIT dimension TO values|position
```

values は 1 つ以上のディメンション・メンバーです。*position* は、メンバーの順序を指定する数値 (1、2 など) か、またはキーワード (FIRST、LAST など) です。

注意： ディメンション・メンバーが整数値の場合は、それらを LIMIT 構文で正しく指定するように注意してください。TEXT データ型のディメンションの場合、'1' (引用符あり) は値が 1 であるメンバーを識別し、1 (引用符なし) はディメンション・リストの最初のメンバーを識別します。

変数のディメンションを制限することにより、後続のコマンドで使用するセルの数が制限されます。

ALL_DESCRIPTIONS の内容を表示するには、次のような OLAP DML コマンドを発行します。

```
LIMIT all_languages TO 1
REPORT W 65 DOWN all_objects w 20 ACROSS all_desctypes: all_descriptions
```

CONCAT ディメンションを制限する方法の 1 つは、ベース・ディメンションを制限してから、CONCAT ディメンションをベース・ディメンションのステータスに制限することです。たとえば、最初の 2 つのディメンションの説明のみを表示するには、次のようなコマンドを発行します。

```
LIMIT all_dimensions TO FIRST 2
LIMIT all_objects TO all_dimensions
LIMIT all_languages TO 1
REPORT W 65 DOWN all_objects w 20 ACROSS all_desctypes: all_descriptions
```

特定のセルの指定

LIMIT コマンドを使用して変数を 1 つの有効なセルに制限することは可能ですが、この目的には修飾データ参照 (QDR) を使用する方が一般的です。QDR は、ディメンションの現在のステータスとは関係なく機能し、コマンドの間のみ有効になります。QDR でディメンションを省略した場合は、ステータスが設定されている最初の値が使用されます。そのため、多次元変数については、LIMIT を使用して QDR の構文を簡略化することができます。

QDR の構文は次のとおりです。

```
variable(dimension 'member', dimension 'member'...)
```

たとえば、次のコマンドは UNITS_CUBE の簡単な説明を表示します。

```
LIMIT all_measures TO 'GLOBAL_AW.UNITS_CUBE.UNITS.MEASURE'
LIMIT all_objects TO all_measures
report all_descriptions(all_desctypes 'SHORT', all_languages 'AMERICAN_AMERICA')
```


変数への値の代入

ディメンション・メンバーを制限することは、変数の値を手動で設定する場合に特に重要です。代入演算子(=)を使用すると、現在選択されているセルに式の値を代入できます。

新しいメジャーの説明を追加するには、次のようなコマンドを使用します。

```
LIMIT all_languages TO 1
LIMIT all_measures TO 'detailed measure name'
LIMIT all_objects TO all_measures
all_descriptions(all_desctypes, 'SHORT')= 'short description'
all_descriptions(all_desctypes, 'LONG')= 'long description'
all_descriptions(all_desctypes, 'PLURAL')= 'plural description'
```

再度 REPORT コマンドを発行して変更が正しく行われていることを確認してから、次のコマンドを発行して変更内容を保存します。

```
UPDATE; COMMIT
```

AW_NAMES 変数

AW_NAMES 変数は、アナリティック・ワークスペース内のオブジェクトのフルネームを識別します。フルネームは、ALL_OBJECTS ディメンションの詳細な名前に対応します。ワークスペース・オブジェクトのフルネームは、次のような書式になります。

```
schema.workspace!object
```

例: GLOBAL_AW.GLOBAL!SALES

AW_NAMES の内容を表示するには、次の OLAP DML コマンドを発行します。

```
REPORT W 60 DOWN all_objects W 35 aw_names
```

新しいメジャーのワークスペース名を追加するには、次のようなコマンドを使用します。

```
LIMIT all_measures TO 'detailed measure name'
LIMIT all_objects TO all_measures
aw_names = 'full workspace object name'
```

再度 REPORT コマンドを発行して変更が正しく行われていることを確認してから、次のコマンドを発行して変更内容を保存します。

```
UPDATE; COMMIT
```

CUBE_MEASURES 値セット

CUBE_MEASURES **値セット**は、各キューブのメジャーを識別します。これは、ALL_CUBES によってディメンション化されており、ALL_MEASURES ディメンションの値を格納しています。

VALUES ファンクションは、値セットの内容を戻します。CUBE_MEASURES の内容を表示するには、次の OLAP DML コマンドを発行します。

```
REPORT W 35 DOWN all_cubes W 55 VALUES (cube_measures)
```

既存のキューブにメジャーを追加するには、次のようなコマンドを使用します。

```
LIMIT all_cubes TO cube
LIMIT cube_measures ADD 'detailed measure name'
```

次に例を示します。

```
LIMIT all_cubes TO 'UNITS_CUBE'
LIMIT cube_measures ADD 'GLOBAL.UNITS_CUBE.PROFIT.MEASURE'
```

再度 REPORT コマンドを発行して変更が正しく行われていることを確認してから、次のコマンドを発行して変更内容を保存します。

```
UPDATE; COMMIT
```

事例 : Global アナリティック・ワークスペースへのメジャーの追加

3-5 ページの「必要とされるビジネス・ファクトの識別」で、Global Corporation で必要なビジネス・メジャーを識別しています。スター・スキーマから取得されたメジャーは、Units、Unit Price および Unit Cost の 3 つのみです。残りのビジネス・メジャーは、この 3 つのメジャーから計算できます。

カスタム・メジャーは、実行時に解決すること、変数に格納することもできます。実行時の計算では、ディスクの記憶域が必要になることも、データ・メンテナンスに必要な処理時間が長くなることもありません。ただし、パフォーマンスは低下する可能性があります。そこで、どのメジャーをその場で計算し、どのメジャーを変数に格納するか（格納する必要がある場合）を決定する必要があります。

必要なビジネス・メジャーの多くは、表 9-1 に示すように、売上、総費用および利益に基づいています。これら 3 つの計算済メジャーは使用頻度が高いので、この例ではこれらを変数に格納します。その他のメジャーは式として実装し、オンデマンドで計算できます。

表 9-1 GLOBAL アナリティック・ワークスペースのカスタム・メジャー

必要なビジネス・メジャー	GLOBAL アナリティック・ワークスペースでのオブジェクト名	式
売上	SALES	UNITS * UNIT_PRICE
総費用	EXTENDED_COST	UNITS * UNIT_COST
利益	MARGIN	SALES - EXTENDED_COST

表 9-1 GLOBAL アナリティック・ワークスペースのカスタム・メジャー

必要なビジネス・メジャー	GLOBAL アナリティック・ワークスペースでのオブジェクト名	式
前の期間（月、四半期、年）からの売上の変化	SALES_PP	LAG(sales, 1, time, LEVELREL time_levelrel)
前の期間からの売上の変化率	SALES_PCTCHG_PP	LAGPCT(sales, 1, time, LEVELREL time_levelrel) * 100
製品シェア	SHARE_SALES_PROD	(sales/sales(product '1')) * 100
チャネル・シェア	SHARE_SALES_CHAN	(sales/sales(channel '1')) * 100
市場シェア	SHARE_SALES_CUST	(sales/sales(customer '1')) * 100
前の期間からの総利益の変化	MARGIN_PP	LAG(margin, 1, time, LEVELREL time_levelrel)
前の期間からの総利益の変化率	MARGIN_PCTCHG_PP	LAGPCT(margin, 1, time, LEVELREL time_levelrel) * 100
製品の総売上に対する総利益の割合	MARGIN_PCT_SALES	(margin/sales(product '1')) * 100
前の期間からの販売単位の変化	UNITS_PP	LAG(units, 1, time, LEVELREL time_levelrel)
単位当たりの利益	UNIT_MARGIN	margin/units

SALES、EXTENDED_COST および MARGIN のメジャーの作成

Sales、Extended Cost および Margin の変数は Units と同じディメンションを持ち、Units キューブに追加されます。

GLOBAL への新規変数の作成

次の手順を実行して、SALES_VARIABLE を作成します。

1. Analytic Workspace Manager のオブジェクト・ビューで、GLOBAL アナリティック・ワークスペースの「Variables」フォルダを開きます。
2. 「UNITS_VARIABLE」を右クリックし、メニューから「**Create Like**」を選択します。
「Create Like」ダイアログ・ボックスが表示されます。
3. 「Destination Name」ボックスに「SALES_VARIABLE」と入力し、「**OK**」をクリックします。
「SALES_VARIABLE」が「Variables」フォルダのリストに追加されます。

4. 「SALES_VARIABLE」をクリックして、この変数をプロパティ・ビューアに表示します。「Properties」ページで、設定を次のように変更します。
AW\$PARENT_NAME: 「SALES」に変更します。
AW\$SEGWDTH_CMD: 変数名を「SALES_VARIABLE」に変更します。
「Apply」をクリックしてプロパティ・ページへの変更を保存します。
5. EXTENDED_COST_VARIABLE および MARGIN_VARIABLE に対しても同じ手順を繰り返します。
6. 新しい定義を保存するには、「File」メニューから「Save」を選択します。

値の計算と変数への格納

次のコマンドは、新しい変数を個別に集計できるようにするために、ベース・レベルでのみデータを計算します。ACROSS コマンドは、ステータスが設定されているディメンション・メンバーに対して繰り返し実行されます。

行末のハイフンは、コマンドが次行に続くことを示します。

注意： SALES_VARIABLE、EXTENDED_COST_VARIABLE および MARGIN_VARIABLE のデータは、ソース変数をリフレッシュするたびに手動でリフレッシュする必要があります。次のようなコマンドは、OLAP DML プログラムにコピーしてリフレッシュ・プロセスの一部として実行できます。

```
" Select base level dimension members
LIMIT time TO time_levelrel 'Month'
LIMIT channel TO channel_levelrel 'CHANNEL'
LIMIT product TO product_levelrel 'ITEM'
LIMIT customer TO customer_levelrel 'SHIP_TO'

" Populate variables using calculations
ACROSS time units_cube_composite DO -
    'extended_cost_variable = units_variable * unit_cost_variable'
ACROSS time units_cube_composite DO -
    'sales_variable = units_variable * unit_price_variable'
ACROSS time units_cube_composite DO -
    'margin_variable = sales_variable - extended_cost_variable'

" Save the new variables
UPDATE
COMMIT
```

メジャー式の作成

次の手順を実行して、SALES 式を作成および登録します。EXTENDED_COST および MARGIN に対しても同じ手順を繰り返します。

1. Analytic Workspace Manager のオブジェクト・ビューで、GLOBAL アナリティック・ワークスペースの「Formulas」フォルダを開きます。
2. 「UNITS」を右クリックし、メニューから「**Create Like**」を選択します。
「Create Like」ダイアログ・ボックスが表示されます。
3. 「Destination Name」ボックスに「SALES」と入力し、「**OK**」をクリックします。
「SALES」が「Formulas」フォルダのリストに追加されます。
4. 「SALES」をクリックし、プロパティ・ページで次の変更を加えます。
「Expression」ページで、AGGREGATE ファンクション・コールの「UNITS_VARIABLE」を「SALES_VARIABLE」に変更します。
「Properties」ページで、AW\$LOGICAL_NAME および SOURCE_NAME の値を「SALES」に変更します。
「**Apply**」をクリックしてプロパティ・ページへの変更を保存します。
5. 新しい定義を保存するには、「File」メニューから「**Save**」を選択します。
6. SALES メジャーを登録するには、OLAP Worksheet を開いて次のコマンドを発行します。

```
" Add SALES to the ALL_MEASURES dimension
MAINTAIN ALL_MEASURES ADD 'global_aw.units_cube.sales.measure'

" Add descriptions to the ALL_DESCRIPTIONS variable
LIMIT all_measures TO 'global_aw.units_cube.sales.measure'
LIMIT all_objects TO all_measures
LIMIT all_languages TO 1
all_descriptions(all_desctypes, 'SHORT')= 'Sales'
all_descriptions(all_desctypes, 'LONG')= 'Sales as Units * Price'
all_descriptions(all_desctypes, 'PLURAL')= 'Sales'

" Add measure name to the AW_NAMES variable

aw_names = 'GLOBAL_AW.GLOBAL!SALES'

" Add measure to the CUBE_MEASURES valueset
LIMIT all_cubes TO 'GLOBAL_AW.UNITS_CUBE.CUBE'
LIMIT cube_measures ADD 'GLOBAL_AW.UNITS_CUBE.SALES.MEASURE'

" Save these changes
UPDATE
```

COMMIT

新規 Global 変数の集計

スタンダード・フォームのメジャーを作成したら、そのメジャーをその他のメジャーと同じように集計できます。新しいメジャーは既存のキューブに追加されているので、新しいメジャー用に既存の集計プランに変更を加えることも、新しい集計プランを作成することもできます。次の手順を実行します。

1. OLAP カタログ・ビューで、GLOBAL アナリティック・ワークスペースの「UNITS_CUBE」が表示されるまで「Cubes」フォルダを開きます。
2. 既存の集計プランに変更を加えるには、次の手順を実行します。
 - a. 「UNITS_CUBE」以下にある「Aggregation Plans」フォルダを開き、目的のプランを右クリックします。
 - b. メニューから「Edit」を選択します。
 - c. SALES、EXTENDED_COST および MARGIN をそのプランに追加します。

または

新しい集計プランを作成するには、「UNITS_CUBE」を右クリックして「**Create Aggregation Plan Using Wizard**」を選択します。ウィザードの各手順を実行します。「Help」をクリックすると、詳細な情報を参照できます。

3. 集計プランをデプロイするには、集計プランを右クリックし、メニューから「**Deploy Aggregation Plan**」を選択します。
4. 「File」メニューから「**Save**」を選択します。

GLOBAL へのその他のカスタム・メジャーの追加

残りのメジャーは、利用可能な任意の方法で実行時に計算できます。次の手順では、この章で説明した方法を使用して新しい式オブジェクトを作成し、これをメジャーとして登録します。別の方法として、DBMS_AW_UTILITIES パッケージを使用して永続的なカスタム・メジャーを定義することもできます（第7章を参照）。

SALES_PP メジャーを定義するには、次の手順を実行します。

1. オブジェクト・ビューで、式の「SALES」を右クリックし、メニューから「**Create Like**」を選択します。
2. 「Create Like」ダイアログ・ボックスの「Destination Name」ボックスに「SALES_PP」と入力します。
3. 新たに作成された式「SALES_PP」をクリックし、プロパティ・ページで次の変更を加えます。

「Properties」ページで、AW\$LOGICAL_NAME および SOURCE_NAME の値を「SALES_PP」に変更します。

「Expression」ページで、AGGREGATE ファンクションを次の LAG ファンクションに置き換えます。

```
LAG(sales, 1, time, LEVELREL time_levelrel)
```

4. 他のメジャーと同じように SALES_PP を登録します。

表 9-1 の他のメジャーに対しても同じ手順を繰り返します。

OLAP DML プログラムによる GLOBAL へのメジャーの追加

前の例では、Analytic Workspace Manager を使用して手動でメジャーを定義する方法を示しました。別のオプションとして OLAP DML プログラムを使用できます。例 9-3 は、メジャーを追加するサンプル・プログラムです。このプログラムは、次の 3 つの引数を取ります。

- メジャーの名前
- ソース変数の名前
- メジャーのキューブの名前

このプログラムは、次のコマンドで実行します。

```
CALL create_measure('display_name' 'source_variable')
```

例: CALL create_measure('Sales' 'sales_variable')

その他の情報はすべて、プログラム冒頭のローカル変数で指定されています。このプログラムを実際のアナリティック・ワークスペースにメジャーを作成するためのテンプレートとして使用する場合は、これらのローカル変数の設定を変更するか、またはコマンドライン引数の変数を変更します。

このプログラムの基本的な処理の流れは次のとおりです。

- ソース変数が存在しているかどうかをチェックし、存在していなければ作成します。プログラムは、変数を移入することはありません。オブジェクト定義を作成するだけです。
- 方程式に AGGREGATE ファンクションを含む式を作成します。方程式は、いつでも変更できます。
- 新しいメジャーをデータベース・スタンダード・フォームのカタログに登録します。

プログラムには、処理内容を理解しやすくするためにコメントが記述されています。また、次の記号も使用されています。

- " (二重引用符) コメントの始点または終点を表します。
- ' (一重引用符) リテラル・テキストを囲みます。
- & (アンパサンド) 式自体のかわりに式の値を使用します。

- ¥ (円記号) 次の文字をコマンド構文の一部ではなくリテラルとして識別します。
- = (等号) 左側の変数を右側の式の値に設定します。
- (ハイフン) コマンドが次行に続くことを示します。
- : (コロン) 処理をリダイレクトするために使用するラベルの名前の後ろに付けます。

例 9-3 メジャーを UNITS_CUBE に追加する DML プログラム

```
DEFINE CREATE_MEASURE PROGRAM
PROGRAM
ARG _displayname      text
ARG _measvar          text
ARG _cube             text
VARIABLE _schema      text
VARIABLE _aw          text
VARIABLE _fullname    text
VARIABLE _measure     text
VARIABLE _datatype    text
VARIABLE _dims        text
VARIABLE _segwidth    text
VARIABLE _aggmap      text
VARIABLE _createdby   text
VARIABLE _fullmeas    text

TRAP ON OOPS          "Redirect processing on error to OOPS label

" Check for measure name argument on command line
IF _displayname EQ na
  THEN SIGNAL noarg 'You must supply a measure name.'
  ELSE _measure = UPCASE(_displayname)

IF _measvar EQ na
  THEN _measvar = JOINCHARS(_measure, '_VARIABLE')

IF _cube EQ na
  THEN _cube = 'UNITS_CUBE'
  ELSE _cube = UPCASE(_cube)

" Change these local variables for your data
_schema = 'GLOBAL_AW'          " Name of the schema that owns the analytic workspace
_aw = 'GLOBAL'                " Name of the analytic workspace
_segwidth = '85 1000000'      " Segment size appropriate for measures in this cube
_aggmap = JOINCHARS(_cube '_AGGMAP_AWCREATEDDEFAULT_1') " Name of default aggmap for cube
_datatype = 'DECIMAL'

_createdby = 'AW$CREATE'
_fullname = UPCASE(JOINCHARS(_schema, '.', _cube, '.', _measvar))
```



```

_dims = OBJ(PROPERTY, 'SYS_DIMS', _cube)
_fullmeas = UPCASE(JOINCHARS(_schema, '.', _cube, '.', _measure, '.MEASURE'))

" Redirect processing to FORMULA label if variable already exists
IF EXISTS(_measvar)
  THEN GOTO FORMULA

" Define the variable
&JOINCHARS('DEFINE ', _measvar, ' VARIABLE ', _datatype, '<', _dims, '>')

" Set Database Standard Form metadata required to register a measure variable
&JOINCHARS('CONSIDER ', _measvar)
PROPERTY 'AW$CLASS' 'EXTENSION'
PROPERTY 'AW$CREATEDBY' _createdby
PROPERTY 'AW$LASTMODIFIED' JOINCHARS(today, '_', tod)
PROPERTY 'AW$PARENT_NAME' _measure
PROPERTY 'AW$ROLE' 'VARIABLE'
PROPERTY 'AW$STATE' 'CREATED'
PROPERTY 'AW$SEGWIDTH_CMD' JOINCHARS(-
  'CHGDFN ', _schema, '.', _aw, '!', _measure, ' SEGWIDTH ', _segwidth)

FORMULA:
" Check if the measure is already defined
IF EXISTS(_measure)
  THEN SIGNAL measexists JOINCHARS(_measure ' already exists.')

" Create the formula
&JOINCHARS('DEFINE ', _measure, ' FORMULA ', _measvar)
" Define the calculation equation
&JOINCHARS('EQ AGGREGATE(', _schema, '.', _aw, '!', _measvar, ' USING ', _aggmap, ')')

" Set properties needed by the OLAP API enablement process
&JOINCHARS('CONSIDER ', _measure)
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$COMPSPEC' _aggmap
PROPERTY 'AW$CREATEDBY' _createdby
PROPERTY 'AW$LASTMODIFIED' JOINCHARS(TODAY, '_', TOD)
PROPERTY 'AW$LOGICAL_NAME' _measure
PROPERTY 'AW$PARENT_NAME' _cube
PROPERTY 'AW$ROLE' 'MEASUREDEF'
PROPERTY 'AW$STATE' 'CREATED'

" Register measure in standard form catalogs
&JOINCHARS('MAINTAIN all_measures ADD ', _measvar, _fullmeas, _measvar)
&JOINCHARS('all_descriptions(all_objects %<ALL_MEASURES: ', _fullmeas, '%>' all_desctypes %SHORT%')
= %', _displayname, '%')
&JOINCHARS('aw_names(all_objects %<ALL_MEASURES: ', _fullmeas, '%>') = %', _schema, '.', _aw, '!',
  _measure, '%')

```

```
&JOINCHARS('LIMIT all_cubes to ¥', _schema, '.', _cube, '.CUBE¥')
&JOINCHARS('LIMIT cube_measures add ' '¥', _fullmeas, '¥')
RETURN
```

```
OOPS:
show 'Program ended in an error.'
END
```

将来の業績の予測

この章では、予測を生成するためにアナリティック・ワークスペースで利用可能なツールについて説明します。予測をスタンダード・フォームのメジャーに格納する方法、および予測結果用のスタンダード・フォームのキューブを作成する方法について説明します。

この章では、次の項目について説明します。

- [予測の作成](#)
- [予測プログラムの開発](#)
- [新規キューブの定義](#)
- [事例 : Global の Sales の予測](#)

予測の作成

OLAP DML は、単純な線形回帰、複数の非線形回帰法、単一指数平滑法、二重指数平滑法および Holt-Winters 法をサポートしています。どの手法を使用すべきかわからない場合は、過去の業績に基づいて、実際のデータに最適な手法を OLAP エンジンに判断させることができます。

ほとんどの予測は、ベース・レベルで計算されます。その後、ベース・レベルの予測データを集計して予測集計を生成します。通常、実際のデータの集計から予測集計を生成することはありません。この章の例は、読者がこの方法で予測集計を生成する予定であることを前提としています。

ただし、集計レベルで予測を生成し、そのデータを下位のレベルに割り当てたいという場合もあります。この方法による予測もサポートされています。

予測の作成手順

次に、予測を作成する手順を示します。各手順は、以降の項で詳細に説明します。

1. 予測対象の期間が時間ディメンションに作成されていることを確認します。必要に応じて期間を追加します。

2. 結果を格納するための変数を定義します。
3. 予測を生成するプログラムを記述します。
4. プログラムをコンパイルおよび実行します。
5. 結果を確認します。
6. 結果のメジャーをキューブに追加します。必要に応じて、予測結果用の新しいキューブを先に作成しておきます。
7. 新しい集計プランを作成するか、または既存の集計プランに変更を加えて、予測結果を格納するメジャーを含めます。集計プランをデプロイします。
8. アプリケーションからアナリティック・ワークスペースを使用できるようにします。

予測期間の作成

予測対象の将来の期間は、アナリティック・ワークスペース内の時間ディメンションのメンバーとして定義する必要があります。期間が定義されていない場合は、次の手順を実行する必要があります。

1. 新しいメンバーおよびその属性を、ソース・スキーマ内の **Time** ディメンション表に追加します。
2. **Analytic Workspace Manager** のリフレッシュ・ウィザードを使用して、アナリティック・ワークスペース内のディメンションに新しいメンバーを追加します。

実際のデータをロードする際は、これらの **Time** ディメンション・メンバーが一致することが保証される方法を使用する必要があります。

結果を格納する変数の定義

予測では、結果を格納するための変数が少なくとも 1 つ必要になります。季節予測および平滑季節予測を求める場合は、最大 3 つの変数が必要になります。通常、これらの変数は、予測を生成するために使用された変数と同じディメンションおよびデータ型を持ちます。

予測を格納する変数を定義するには、次の手順を実行します。

1. 結果の変数を、スタンダード・フォームのメジャーとして定義します。
変数および集計式を定義する方法と、メジャーを登録する方法については、9-8 ページの「[キューブへのカスタム・メジャーの追加](#)」を参照してください。
2. 季節予測を求める場合は、季節係数を格納するための第 2 の変数を定義します。この変数には、スタンダード・フォームのプロパティを割り当てないでください。かわりに、次の手順を実行します。
 - a. オブジェクト・ビューで、目的のアナリティック・ワークスペースのフォルダを開きます。

- b. 「Variables」を右クリックし、メニューから「**Create Variable**」を選択します。
 - c. 変数を DECIMAL データ型として定義します。
 - d. 「Dimensions」ページで、キューブ内の変数のディメンションを適切な順序でリストします。通常は、**Time** を先頭に、その後にコンポジット・ディメンションをリストします。
3. 平滑季節予測を求める場合は、平滑係数を格納するための第 3 の変数を定義します。季節係数の変数を右クリックして「**Create Like**」を選択し、この変数をコピーします。

予測プログラムの開発

予測では、いくつかの関連コマンドを使用します。これらは、必ず OLAP DML プログラムから実行されます。これらのコマンドは、**予測コンテキスト**を定義します。次のコマンドを記載されているとおりの順序で使用します。

1. FCOPEN ファンクション。予測コンテキストを開き、そのハンドルを戻します。
2. FCSET コマンド。予測の特性を指定します。
3. FCEXEC コマンド。予測を実行し、予測データを Oracle OLAP 変数に移入します。
4. FCQUERY ファンクション（オプション）。予測の特性または予測の試行に関する情報を取得します。
5. FCCLOSE コマンド。予測コンテキストを閉じます。

例 10-1 は、これらのコマンド、および予測で一般的に使用されるその他のコマンドのテンプレートです。

例 10-1 予測のテンプレート

```
VARIABLE handle INTEGER      " Define a local variable
TRAP ON OOPS                 " Redirect processing on error to OOPS label

" Select base level time periods
LIMIT time_dim TO levelrel_time 'base_data'
" Keep historical and forecast periods
LIMIT time_dim KEEP LAST n

" Open a handle for the forecast
handle = FCOPEN('forecast_name')
" Specify the forecast method
FCSET handle METHOD 'method' descriptors
" Execute the forecast and identify source and target variables
FCEXEC handle TIME time_dim INTO target_var1 SEASONAL -
      target_var2 SMSEASONAL target_var3 source_var
FCCLOSE handle               " Close the forecast
```

```
RETURN  
  
OOPS:  
SHOW 'Error running program'
```

予測の生成

予測データを生成するには、次のようなコマンドを使用して予測プログラムを実行します。

```
CALL forecast_sales
```

新規キューブの定義

キューブは、同じような特性のメジャーをまとめる手段を提供します。メジャーは、実質的にいくつでも特定のキューブに関連付けることができます。ただし、一部の計算済メジャーについては、同様の特性を持つキューブがすでにある場合でも、個別にキューブを作成した方がよい場合もあります。たとえば、予測メジャーを、実績メジャーを持つ既存のキューブに追加できる場合でも、両者を混同してしまわないように別のキューブに追加した方がよい場合もあります。

キューブのメタデータには、そのメジャーのソース変数に関する情報（コンポジットや集計マップの名前など）が含まれます。以降の説明は、これらのオブジェクトがすでに存在していることを前提としています。これらのオブジェクトがまだない場合は、[第6章](#)を参照してください。

Cubedef オブジェクトの作成

cubedef オブジェクトは、キューブのディメンションの名前を示すテキスト・ディメンションです（8-21 ページの「[スタンダード・フォームのキューブ](#)」を参照）。Analytic Workspace Manager を使用して *cubedef* オブジェクトを作成するには、次の手順を実行します。

1. オブジェクト・ビューを開き、目的のアナリティック・ワークスペースのフォルダを開きます。
2. 「Dimension」 フォルダを開き、既存のキューブの *cubedef* ディメンションを右クリックします。
3. メニューから「**Create Like**」を選択します。
「Create Like」 ダイアログ・ボックスが表示されます。
4. 新しいキューブの名前を入力します。

ワークスペース内の他のキューブの名前と揃えるため、名前の末尾には必ず「_CUBE」を付ける必要があります（「SALES_CUBE」など）。

5. 新しい *cubedef* ディメンションを「Dimension」フォルダから選択し、プロパティ・ビューアで次の変更を加えます。
 - 「Basic」ページ：新しい説明を入力します。
 - 「Properties」ページ：AW\$LOADPRGS、LOAD_TYPE、SOURCE_NAME および SOURCE_OWNER の各プロパティを削除します。次に、その他のプロパティの値を編集して、新しいオブジェクトに適切な値に変更します。これらのプロパティの詳細は、8-22 ページの表 8-13 を参照してください。

新しいキューブではロード・プログラムを使用してデータを取得することはないので、リフレッシュ・ウィザードにこのキューブが表示されないようにします。
6. OLAP Worksheet を開きます。次のようなコマンドを使用して、ディメンションの名前を値として追加します。


```
MAINTAIN cube ADD 'dimension' 'dimension' ...
```


 例:

```
MAINTAIN new_cube ADD 'PRODUCT' 'TIME'
```
7. 「File」メニューから「Save」を選択して、アナリティック・ワークスペースおよび現在のスキーマ内のすべてのオブジェクトを更新します。

デフォルト集計マップの作成

すべてのキューブには、デフォルト集計マップが必要です。デフォルト集計マップは、初期状態で、すべての問合せに対する回答が完全に解決されたメジャーによって行われるようにするために使用されます。集計マップ（OLAP DML の用語では「aggmap オブジェクト」）には、集計に必要な規則がすべて含まれます。

キューブのデフォルト集計マップを作成するには、次の手順を実行します。

1. オブジェクト・ビューで、「Aggregation Maps」フォルダを開きます。
2. 同じような特性を持つキューブのデフォルト集計マップを右クリックし、メニューから「Create Like」を選択します。

デフォルト集計マップは、UNITS_CUBE_AGGMAP_AWCREATEDDEFAULT_1 のような名前を持ちます。

3. 新しい集計マップにこのような名前を付けます。たとえば、FORECAST_CUBE_AGGMAP_AWCREATEDDEFAULT_1 のようにします。
4. 新しい集計マップを選択し、次の変更を加えます。
 - 「Properties」ページ：AW\$PARENT_NAME の値を新しいキューブの名前に変更します。
 - 「Aggmap」ページ：新しいキューブの各ディメンションに RELATION コマンドが 1 つずつ（1 つだけ）あることを確認します。必要に応じて変更を行い、「Compile」をクリックして変更後の構文が正しいかどうかをチェックします。

5. 「Apply」をクリックして、これらの変更を現在のセッションに保存します。
6. 「File」メニューから「Save」を選択して、将来のセッションで利用できるようにこれらの変更を保存します。

新規キューブの登録

キューブの登録は、メジャーの登録（9-11 ページの「[新規メジャーの登録](#)」を参照）とよく似ています。キューブの登録では、メジャーの登録とほとんど同じカタログが関係します。そのプロパティ・シートは、Analytic Workspace Manager で確認できます。または、OLAP Worksheet で次のコマンドを発行して、その定義を表示することもできます。

```
DESCRIBE all_cubes all_descriptions aw_names cube_measures
```

これらのカタログの詳細は、8-26 ページの「[スタンダード・フォームのカタログ](#)」を参照してください。

ALL_CUBES ディメンションへのキューブの追加

ALL_CUBES ディメンションは、アナリティック・ワークスペースにあるすべてのキューブのリストです。その内容を表示するには、次の OLAP DML コマンドを発行します。

```
REPORT W 40 all_cubes
```

キューブの名前は、次のような詳細な形式になります。

```
schema.cube.CUBE
```

新しいキューブを ALL_CUBES に追加するには、次のコマンド構文を使用します。

```
MAINTAIN all_cubes ADD detailed_cube_name
```

次に例を示します。

```
MAINTAIN all_cubes ADD 'GLOBAL.ANALYTICS_CUBE.CUBE'
```

再度 REPORT コマンドを発行して変更が正しく行われていることを確認してから、次のコマンドを発行して変更内容を保存します。

```
UPDATE; COMMIT
```

ALL_DESCRIPTIONS 変数へのキューブの追加

ALL_DESCRIPTIONS 変数は、各オブジェクトの簡単な説明、詳細な説明および複数説明を格納します（9-13 ページの「[ALL_DESCRIPTIONS 変数](#)」を参照）。

新しいキューブの説明を追加するには、次のようなコマンドを使用します。

```
LIMIT all_languages TO 'AMERICAN.AMERICA'  
LIMIT all_cubes TO 'detailed_mcubename'
```



```
LIMIT all_objects TO all_cubes  
all_descriptions(all_desctypes, 'SHORT')= 'short description'  
all_descriptions(all_desctypes, 'LONG')= 'long description'  
all_descriptions(all_desctypes, 'PLURAL')= 'plural description'
```

再度 REPORT コマンドを発行して変更が正しく行われていることを確認してから、次のコマンドを発行して変更内容を保存します。

```
UPDATE; COMMIT
```

AW_NAMES 変数へのキューブの追加

AW_NAMES 変数は、アナリティック・ワークスペース内のオブジェクトのフルネームを識別します (9-15 ページの「[AW_NAMES 変数](#)」を参照)。

新しいキューブのワークスペース名を追加するには、次のようなコマンドを使用します。

```
LIMIT all_cubes TO 'detailed_cube_name'  
LIMIT all_objects TO all_cubes  
aw_names = 'full workspace object name'
```

再度 REPORT コマンドを発行して変更が正しく行われていることを確認してから、次のコマンドを発行して変更内容を保存します。

```
UPDATE; COMMIT
```

CUBE_MEASURES 値セット内の新規キューブへのメジャーの追加

CUBE_MEASURES [値セット](#)は、各キューブのメジャーを識別します (9-15 ページの「[CUBE_MEASURES 値セット](#)」を参照)。

新しいキューブにメジャーを追加するには、次のようなコマンドを使用します。

```
LIMIT all_cubes TO cube  
LIMIT cube_measures ADD 'detailed measure name . . .'
```

次に例を示します。

```
LIMIT all_cubes TO 'ANALYTICS_CUBE'  
LIMIT cube_measures ADD 'GLOBAL.ANALYTICS_CUBE.PROFIT.MEASURE' -  
    'GLOBAL.ANALYTICS_CUBE.SALES_PCTCHG.MEASURE'
```

再度 REPORT コマンドを発行して変更が正しく行われていることを確認してから、次のコマンドを発行して変更内容を保存します。

```
UPDATE; COMMIT
```

手動で作成したキューブに関するトラブルシューティング

キューブの作成手順に誤りがあると、キューブを集計したり、アナリティック・ワークスペースをリフレッシュまたは有効化したりする際にエラーが発生します。失敗の原因を特定するには、次のチェック・リストの作業を実行します。

- 8-21 ページの「[スタンダード・フォームのキューブ](#)」に示されているプロパティを参照しながら、キューブ・ディメンションのプロパティをチェックします。別のキューブをコピーした場合は、必要なすべての変更をプロパティ値に加えてあることを確認します。

- キューブ・ディメンションを移入してあることを確認します。

```
REPORT forecast_cube
```

```
FORECAST_CUBE
```

```
-----
```

```
CHANNEL
```

```
CUSTOMER
```

```
PRODUCT
```

```
TIME
```

- キューブを ALL_CUBES ディメンションに追加してあることを確認します。

```
REPORT W 30 all_cubes
```

```
ALL_CUBES
```

```
-----
```

```
GLOBAL_AW.PRICE_CUBE.CUBE
```

```
GLOBAL_AW.UNITS_CUBE.CUBE
```

```
GLOBAL_AW.FORECAST_CUBE.CUBE
```

- キューブを AW_NAMES 変数に追加してあることを確認します。

```
LIMIT all_objects TO all_cubes
```

```
REPORT W 42 DOWN all_objects W 35 aw_names
```

```
ALL_OBJECTS
```

```
AW_NAMES
```

```
-----
```

```
<ALL_CUBES: GLOBAL_AW.PRICE_CUBE.CUBE> GLOBAL_AW.GLOBAL!PRICE_CUBE
```

```
<ALL_CUBES: GLOBAL_AW.UNITS_CUBE.CUBE> GLOBAL_AW.GLOBAL!UNITS_CUBE
```

```
<ALL_CUBES: GLOBAL_AW.FORECAST_CUBE.CUBE> GLOBAL_AW.GLOBAL!FORECAST_CUBE
```

事例 : Global の Sales の予測

実績データを格納する Units キューブに予測メジャーを追加することはできましたが、この例では、予測メジャー用に FORECAST_CUBE という新しいキューブを作成します。

FORECAST_CUBE は、UNITS_CUBE と同じディメンションを持ちます。したがってこの 2 つ

のキューブは、コンポジット・ディメンション UNITS_CUBE_COMPOSITE を共有することになります。予測により、Forecast キューブに 1 つのメジャーが移入されます。

この例は、第 9 章で説明した SALES メジャーが作成済であることを前提としています。

予測メジャー用の新規キューブの定義

FORECAST_CUBE という新しいキューブを作成するための基本的な手順は、次のとおりです。

1. Analytic Workspace Manager のオブジェクト・ビューで、「Create Like」を使用して UNITS_CUBE を FORECAST_CUBE としてコピーします。この手順では、UNITS_CUBE のオブジェクト定義はコピーされますが、その内容はコピーされません。
2. FORECAST_CUBE の「Properties」ページで次のプロパティを変更し、「Apply」をクリックします。
 - AW\$LOADPRGS、LOAD_TYPE、SOURCE_NAME、SOURCE_OWNER: リフレッシュ・ウィザードがキューブを無視するように、これらのプロパティを削除します。
 - AW\$LOGICAL_NAME: 「FORECAST_CUBE」に設定します。
 - AGGMAPLIST: 「FORECAST_CUBE_AGGMAP_AWCREATEDDEFAULT_1」に設定します。
 - DISPLAY_NAME: 「Sales Forecast Cube」に設定します。
3. 「UNITS_CUBE_AGGMAP_AWCREATEDDEFAULT_1」を右クリックして「Create Like」を選択し、「FORECAST_CUBE_AGGMAP_AWCREATEDDEFAULT_1」という名前のデフォルト集計マップを作成します。
4. 「Properties」ページで、AW\$PARENT_NAME の値を「FORECAST_CUBE」に変更します。
新しいキューブは、Units キューブと同じディメンションを持ちます。ディメンションが異なる場合は、集計マップを編集する必要があります。
5. 新しい定義を保存するには、「File」メニューから「Save」を選択します。
6. Forecast キューブのディメンションを追加するには、次のコマンドを発行します。

```
MAINTAIN forecast_cube ADD 'CHANNEL' 'CUSTOMER' 'PRODUCT' 'TIME'
```

7. FORECAST_CUBE キューブを登録するには、OLAP Worksheet を開いて次のコマンドを発行します。

```
" Add FORECAST_CUBE to the ALL_CUBES dimension
MAINTAIN ALL_CUBES ADD 'GLOBAL_AW.FORECAST_CUBE.CUBE'

" Add descriptions to the ALL_DESCRIPTIONS variable
LIMIT all_cubes TO 'GLOBAL_AW.FORECAST_CUBE.CUBE'
LIMIT all_objects TO all_cubes
```

```
LIMIT all_languages TO 1
all_descriptions(all_descetypes, 'SHORT')= 'Sales Fcast'
all_descriptions(all_descetypes, 'LONG')= 'Sales Forecast'
all_descriptions(all_descetypes, 'PLURAL')= 'Sales Forecasts'

" Add cube name to the AW_NAMES variable

aw_names = 'GLOBAL_AW.GLOBAL!FORECAST_CUBE'

" Save these changes
UPDATE
COMMIT
```

Global の売上の予測メジャーの定義

この予測の結果は、3 つの変数に格納されます。分析対象となるのは 1 つのみです。残りの 2 つは、予測の作成に使用される季節係数および平滑調整係数を保持します。

スタンダード・フォームのメジャーを定義する最も簡単な方法は、[例 9-3 「メジャーを UNITS_CUBE に追加する DML プログラム」](#) に示した CREATE_MEASURE プログラムを使用することです。次の手順を実行します。

1. Analytic Workspace Manager のオブジェクト・ビューで、GLOBAL を読み込み / 書き込みモードでアタッチします。プログラムが別のワークスペースにある場合は、読み込み専用モードか読み込み / 書き込みモードでそのワークスペースもアタッチする必要があります。
2. OLAP Worksheet を開き、次のコマンドを発行します。

```
AW LIST
```

ここで、GLOBAL アナリティック・ワークスペースがリストの先頭に表示されていなければなりません。新しいワークスペース・オブジェクトは、リストの先頭にあるアナリティック・ワークスペースに作成されるためです。GLOBAL が先頭に表示されていない場合は、次のコマンドを発行します。

```
AW ATTACH global FIRST
```

3. 次のコマンドを発行して、予測結果用のスタンダード・フォームのメジャーを作成します。

```
CALL create_measure('sales_fcast', na, 'forecast_cube')
```

引数で指定しているのは、新しいメジャー (SALES_FCAST)、新しい変数 (名前はメジャーの名前から構成される)、およびキューブ (FORECAST_CUBE) です。

4. 次の手順を実行して、SALES_FCAST_SEASONAL を作成します。
 - a. オブジェクト・ビューで、「Variables」を右クリックし、「**Create Variable**」を選択します。

「Create Variable」ダイアログ・ボックスが表示されます。

- b. 「Basic」 ページで、SALES_FCAST_SEASONAL を DECIMAL データ型として定義します。
 - c. 「Dimensions」 ページで、TIME を先頭に、次に UNITS_CUBE_COMPOSITE をリストします。
 - d. 「Create」 をクリックして、この変数定義を現在のセッションに保存します。
5. 「SALES_FCAST_SEASONAL」 を右クリックしてメニューから「Create Like」を選択し、「SALES_FCAST_SMOOTHED」を作成します。
 6. 「File」 メニューから「Save」を選択します。

Global の売上の予測プログラムの開発

例 10-2 は、GLOBAL アナリティック・ワークスペースの売上を予測する FORECAST_SALES というプログラムです。このプログラムは、実際のアナリティック・ワークスペースの予測プログラムのテンプレートとして使用できます。

予測自体に必要なコマンドは 4 つだけです。デフォルトの予測メソッドは AUTOMATIC です。このメソッドでは、OLAP エンジンがデータに基づいて最適なメソッドを選択します。季節も指定され、季節変数および平滑季節変数の両方が使用されます。

履歴期間と予測期間の識別

GLOBAL アナリティック・ワークスペースには、65 の履歴期間（Jan-98 ～ May-03）と 12 の予測期間（Jun-03 ～ May-04）があります。基本期間は月なので、季節調整は 12 の期間のサイクルに基づいて行われます。このプログラムは、LIMIT ファンクションの INTEGER 引数を使用して最後の履歴期間の位置（数値）を取得し、その位置に応じて TIME のステータスを設定します。

FORECAST_SALES サンプル・プログラムの引数

FORECAST_SALES プログラムは、次の 5 つの引数を取ります。

- 予測メソッド（AUTOMATIC、LINREF、NLREL1 ～ NLREG5、SESMOOTH、DESMOOTH、HOLT/WINTERS）。
- データが存在する最後の期間の詳細な説明。
- 予測で使用する履歴期間の数。
- 予測する期間の数。
- 季節サイクル内の期間の数。

コマンドラインで引数を省略できるように、これらの引数にはデフォルト値が設定されています。次のようなコマンドでプログラムを実行できます。

```
CALL forecast_sales
CALL forecast_sales('holt/winters')
CALL forecast_sales(na, na, 36, 6)
```

引数は順番にプログラムに渡されるので、一部の引数に関しては、3 番目の例に示したようにブレースホルダ値として NA を渡す必要があります。後方の引数は単純に省略できます。

プログラムの引数と事前設定されたローカル変数の一部は、そのステータスのディメンション・メンバーを選択するために使用されます。計算済の集計が予測で使用されないようにするため、すべてのディメンションはベース・レベルに制限されます。また、ソース履歴期間およびターゲット予測期間だけにステータスが設定されるように、TIME ディメンションを制限する必要があります。

例 10-2 Global の売上の予測プログラム

```
DEFINE FORECAST_SALES PROGRAM
PROGRAM
ARG _method          TEXT    " Forecasting method
ARG _last_time        TEXT    " Long desc of last hist time period
ARG _histperiods      INT     " Number of historical periods
ARG _fcast_periods    INT     " Number of forecast periods
ARG _periodicity      INT     " Number of periods in a cycle
VARIABLE _time_level  TEXT    " Base level of time dimension
VARIABLE _channel_level TEXT   " Base level of channel dimension
VARIABLE _product_level TEXT   " Base level of product dimension
VARIABLE _customer_level TEXT  " Base level of customer dimension
VARIABLE _last_time_pos INT    " Numeric position of _last_time in time dim
VARIABLE _handle      INT     " Forecast handle

TRAP ON OOPS          " Divert processing on error to OOPS label

" Set default values for args
if _method eq na
  then _method = 'AUTOMATIC'
if _last_time eq na
  then _last_time = 'May-03'
if _histperiods eq na
  then _histperiods = 48
if _fcast_periods eq na
  then _fcast_periods = 12
if _periodicity eq na
  then _periodicity = 12

" Identify base levels of dimensions
_time_level='MONTH'
_channel_level='CHANNEL'
_product_level= 'ITEM'
```

```

_customer_level='SHIP_TO'

" Set dimension status to base level
PUSH time channel product customer
LIMIT channel TO channel_levelrel EQ _channel_level
LIMIT product TO product_levelrel EQ _product_level
LIMIT customer TO customer_levelrel EQ _customer_level
LIMIT time TO time_levelrel EQ _time_level

" Check time parameters of forecast and refine status of time dimension
_last_time_pos = LIMIT(INTEGER time TO time_long_description EQ _last_time)
IF _histperiods + _fcast_periods GT STATLEN(time)
  THEN SIGNAL toosmall 'You specified more time periods than are defined.'
IF _last_time_pos - _histperiods lt 0
  THEN SIGNAL nohist 'You specified too many historical periods.'
IF _last_time_pos + _fcast_periods GT STATLAST(time)
  THEN SIGNAL nofuture 'You specified too many forecast periods.'
ELSE LIMIT time KEEP -
  (_last_time_pos - _histperiods + 1) TO (_last_time_pos + _fcast_periods)

" Run the forecast
_handle = FCOPEN('sales')
FCSET _handle METHOD _method HISTPERIODS _histperiods PERIODICITY _periodicity
FCEXEC _handle TIME time INTO sales_fcast_variable -
  SEASONAL sales_fcast_seasonal_variable -
  SMSEASONAL sales_fcast_smoothed_variable sales
FCCLOSE _handle

POP time channel product customer
RETURN

OOPS:
SHOW 'Program ended in an error.'
END

```

Global の売上の予測データの検討

例 10-3 は、FORECAST_SALES プログラムをデフォルト設定で実行した結果の抜粋です。SALES メジャーは、履歴期間（May-03 以前）のデータのみを持っています。SALES_FCAST メジャーは、予測期間（Jun-03 以降）のデータのみを持っています。SALES_FCAST_SEASONAL と SALES_FCAST_SMOOTHED は、最初の季節サイクル（12 か月）のセルに係数を格納しています。

非常にスパースなメジャーのデータを特定することは困難な場合があります。時間ディメンションを予測対象の期間に制限し、その他すべてのディメンションはデータがあることがわかっている 1 つのメンバーに制限します。例 10-3 は、ディメンションをレベル、属性値、位置または値で制限する方法も示しています。

例 10-3 Global の売上の予測結果の表示

```
LIMIT time TO time_levelrel EQ 'MONTH'          "Select base level time periods
"Remove periods not used in forecast
LIMIT time REMOVE time_end_date LT '30JUN00'
"Select base level channels and products
LIMIT channel TO channel_long_description EQ 'Direct Sales'
LIMIT product TO product_levelrel EQ 'ITEM'
LIMIT product KEEP FIRST 1                      "Keep just the first product
LIMIT customer TO '51'                          "Select customer 51
REPORT W 5 DOWN time W 12 <time_long_description sales -
      sales_fcast sales_fcast_seasonal sales_fcast_smoothed>
```

CHANNEL: 2
PRODUCT: 13
CUSTOMER: 51
ALL_LANGUAGES: AMERICAN_AMERICA

-----TIME_HIERLIST-----					
-----CALENDAR-----					
TIME	TIME_LONG_DE SCRIPTION	SALES	SALES_FCAST	SALES_FCAST_ SEASONAL	SALES_FCAST_ SMOOTHED
48	Jun-00	2,893.68	NA	0.32	0.70
49	Jul-00	2,840.35	NA	0.78	0.70
50	Aug-00	5,739.92	NA	1.16	0.70
51	Sep-00	5,821.08	NA	0.74	0.70
52	Oct-00	5,034.92	NA	0.32	0.69
53	Nov-00	2,488.27	NA	0.74	1.37
54	Dec-00	5,100.34	NA	1.36	1.22
55	Jan-01	NA	NA	0.70	1.24
56	Feb-01	4,903.58	NA	1.35	1.28
57	Mar-01	4,893.34	NA	1.39	1.32
58	Apr-01	4,824.84	NA	1.49	1.37
59	May-01	4,791.26	NA	1.65	0.70
	.				
	.				
	.				
91	Jun-03	NA	0.98	NA	NA
92	Jul-03	NA	1.01	NA	NA
93	Aug-03	NA	1.21	NA	NA
94	Sep-03	NA	1.12	NA	NA
95	Oct-03	NA	1.07	NA	NA
96	Nov-03	NA	1.05	NA	NA
97	Dec-03	NA	1.06	NA	NA
103	Jan-04	NA	1.79	NA	NA
104	Feb-04	NA	1.84	NA	NA
105	Mar-04	NA	1.89	NA	NA
106	Apr-04	NA	1.93	NA	NA

107	May-04	NA	1.96	NA	NA
108	Jun-04	NA	NA	NA	NA

予測メジャーの集計と有効化

新しい Forecast キューブの集計プランは、次のようにして、その他のキューブと同じように作成およびデプロイできます。

1. OLAP カタログ・ビューで、GLOBAL アナリティック・ワークスペースのフォルダを開きます。
2. 「FORECAST_CUBE」を右クリックし、メニューから「**Create Aggregation Plan Using Wizard**」を選択します。ウィザードの各手順を実行します。
3. 集計プランを作成したら、「Aggregation Plans」フォルダを開きます。
4. 作成した集計プランの名前を右クリックし、「**Deploy Aggregation Plan Using Wizard**」を選択します。

アプリケーションから予測を使用できるようにするために、GLOBAL アナリティック・ワークスペースを再有効化します。

これらのウィザードの実行に問題がある場合は、10-8 ページの「[手動で作成したキューブに関するトラブルシューティング](#)」を参照してください。

他のソースからのデータの取得

Oracle OLAP には、スター・スキーマまたはスノーフレイク・スキーマ以外のソースからスタンダード・フォームのアナリティック・ワークスペースを作成したり、それらのソースから既存のワークスペースにデータを追加したりできるように、データ取得機能が用意されています。この章では、これらの機能について説明します。この章では、次の項目について説明します。

- [OLAP データ取得サブシステムの概要](#)
- [スタンダード・フォームのアナリティック・ワークスペースを手動で作成する方法](#)
- [フラット・ファイルの読み込み](#)
- [リレーショナル表からのデータのフェッチ](#)
- [その他のメタデータ・オブジェクトの移入](#)
- [事歴: 他のソースからの GLOBALX ワークスペースの作成](#)

OLAP データ取得サブシステムの概要

Oracle Warehouse Builder を使用すると、様々なデータソースをスター・スキーマに変換し、そのスター・スキーマからアナリティック・ワークスペースを作成することができます。この他、空のスタンダード・フォームのオブジェクトを格納するアナリティック・ワークスペースを作成し、OLAP DML の機能を使用して、データソースから直接これらのオブジェクトを移入するという方法もあります。

Analytic Workspace Manager または Oracle Warehouse Builder を使用してアナリティック・ワークスペースを作成済であっても、シンジケート・データや政府による企業統計などのソースからメジャーを追加することが必要な場合があります。この場合はまず、[第 8 章「スタンダード・フォームのアナリティック・ワークスペースの詳細」](#) および [第 9 章「スタンダード・フォームのアナリティック・ワークスペースへのメジャーの追加」](#) の説明に従って、スタンダード・フォームのワークスペース・オブジェクトを定義および登録します。そして、この章で説明されているいずれかの方法でオブジェクトを移入します。

この章では、OLAP ツールを使用してスタンダード・フォームのアナリティック・ワークスペースを生成する方法と、OLAP DML を使用して様々なソースから手動でオブジェクトを移入する方法について説明します。

OLAP DML では、次のソースからデータをロードできます。

- **フラット・ファイル**。ファイル読み込みコマンドは、フラット・ファイルからデータをロードします。これにより、スプレッドシート、シンジケート・ソース、政府ソースまたはレガシー・データベース・システムのデータを使用できるようになります。
- **リレーショナル表**。OLAP DML の SQL コマンドを使用すると、アナリティック・ワークスペースからほとんどの SQL コマンドを実行できます。SQL コマンドを使用すると、適切なデータ型のデータを、リレーショナル表またはビューからアナリティック・ワークスペースにフェッチできます。DBMS_AWM パッケージなどの OLAP ツールは、SQL コマンドを使用してアナリティック・ワークスペースを移入します。
- **EIF ファイル**。IMPORT コマンドおよび EXPORT コマンドを使用すると、レガシーの OLAP Server データベースから、スタンダード・フォームのアナリティック・ワークスペースを作成できます。変換プログラムは、Oracle Express Objects のメタデータを格納する OLAP Server データベースに対して使用できます。[付録 A](#) を参照してください。

スタンダード・フォームのアナリティック・ワークスペースを手動で作成する方法

他のソースからスタンダード・フォームのアナリティック・ワークスペースを作成する手順は、基本的にはスター・スキーマまたはスノーフレーク・スキーマから作成する場合の手順と同じです。主な違いは次の 2 点です。

- データ・ウェアハウスのスター・スキーマまたはスノーフレーク・スキーマにデータをロードするのではなく、単純に空の表を作成します。これらの表は、アナリティック・ワークスペースを作成する DBMS_AWM パッケージで必要となる OLAP カタログ・メタデータを定義する際の基盤となります。データは、アナリティック・ワークスペースに直接ロードされるまでは、元の形式が維持されます。
- DBMS_AWM パッケージは、最初のロード時のデータの特性を利用して、ディメンション順序やセグメント・サイズなどのデフォルトの選択を行います。ここでは、適切な選択を行うためのデータは存在しないので、正しい値を指定する必要があります。DBMS_AWM パッケージを直接使用すると、最も多くの要素を制御できます。ただしこの場合でも、必要に応じて Analytic Workspace Manager または Oracle Warehouse Builder を使用して、データをロードする前に結果に変更を加えることができます。

フラット・ファイル、リレーショナル表、または Oracle Express Objects メタデータを持たない OLAP Server データベースからスタンダード・フォームのアナリティック・ワークスペースを生成するには、次の手順を実行します（Oracle Express Objects データベースを変換する場合は、この手順をスキップして[付録 A](#)に進んでください）。

1. 実際のデータのディメンション、属性、メジャーおよびキューブを識別し、この情報を使用してスター・スキーマを設計します。

設計は、紙と鉛筆、データベース設計ソフトウェア・パッケージなど、どのようなものを使用して行ってもかまいません。

2. ディメンション表およびファクト表を作成して、この設計を実装します。

表を作成するには、SQL*Plus で直接 SQL の CREATE TABLE 文を発行するか、または Oracle Enterprise Manager などのグラフィカルなインタフェースを使用します。ここで言うのは、表の移入ではなく、表の作成であることに注意してください。

3. スター・スキーマの OLAP カタログ・メタデータを作成します。

CWM1 メタデータまたは CWM2 メタデータを作成するには、Oracle Enterprise Manager の OLAP 管理ツール、Oracle Warehouse Builder の OLAP Bridge または CWM2 PL/SQL パッケージのいずれかを使用します。

4. OLAP カタログ・メタデータからスタンダード・フォームのアナリティック・ワークスペースを作成します。

Analytic Workspace Manager のアナリティック・ワークスペース作成ウィザード、Oracle Warehouse Builder の OLAP Bridge または DBMS_AWM PL/SQL パッケージのいずれかを使用します。表にデータが格納されていない場合でも、完全ロードを指定します。これは、すべてのカタログがアナリティック・ワークスペースに正しく移入されるようにするためです。

5. アナリティック・ワークスペースを確認し、オブジェクト定義に必要な変更を加えます。特に、コンポジットおよびデータ変数のディメンション順序をチェックし、ターゲット変数に適切なセグメント・サイズを設定します。

この種の変更の例については、11-18 ページの「[事歴: 他のソースからの GLOBALX ワークスペースの作成](#)」を参照してください。

6. スタンダード・フォームのアナリティック・ワークスペースのディメンション、リレーションおよび変数にデータをロードします。

この章で説明されているいずれかの方法を使用します。

7. ワークスペース・メタデータに必要な変更を加えます。

これで、スタンダード・フォームのアナリティック・ワークスペースが作成されました。このワークスペースに対し、スタンダード・フォームのワークスペース用に提供されている任意の集計ツールおよび配置ツールを使用できます。ただし、リフレッシュ用に作成した OLAP DML プログラムを使用して、データをリフレッシュする必要があります。

フラット・ファイルの読み込み

OLAP DML のファイル読み込みコマンドを使用すると、バイナリ、パック 10 進データ、テキストなど、様々な形式の外部ファイルからデータを取得できます。一部のファイル読み込みコ

マンドは個別に使用できますが、最も望ましいのは、ファイル読み込みコマンドをプログラム内に記述することです。これにより、入力ミス を最小限にすることができ、またコマンドを少量のデータ・セットでテストすることができます。またプログラムを使用することで、ファイル内の大量のレコードに対して複数のコマンドを繰り返し実行するといった処理が可能になります。それ以降は、このプログラムをデータのリフレッシュに使用できます。

ファイル読み込みプログラムについて

表 11-1 に、OLAP DML のファイル読み込みコマンドを示します。

表 11-1 OLAP DML のファイル読み込みコマンド

コマンド	説明
FILECLOSE コマンド	開いているファイルを閉じます。
FILEERROR ファンクション	FILEREAD コマンドおよび FILEVIEW コマンドを使用して入力ファイルのレコードを処理しているときに発生した最初のエラーに関する情報を戻します。
FILEGET ファンクション	読み込みのために開かれているファイルのテキストを戻します。
FILENEXT ファンクション	FILEVIEW コマンドでレコードを処理できるようにします。このファンクションは、レコードを読み込み可能な場合は YES を戻し、ファイルの末尾に達した場合は NO を戻します。
FILEOPEN ファンクション	ファイルを開き、ファイル・ユニット番号（任意の整数値）を割り当てて、その番号を戻します。
FILEPUT コマンド	テキスト式で指定されたデータを、WRITE モードまたは APPEND モードで開かれたファイルに書き込みます。
FILEQUERY ファンクション	1 つ以上のファイルに関する情報を戻します。
FILEREAD コマンド	入力ファイルのレコードを読み込み、そのデータを処理します。そして、入力レコードの各フィールドの記述に従い、ワークスペースのディメンション、コンポジット、リレーションおよび変数にそのデータを格納します。
FILESET コマンド	指定されたファイル・ユニットのページング属性を設定します。
FILEVIEW コマンド	FILENEXT ファンクションとともに使用します。入力ファイルのレコードを一度に 1 つずつ読み込み、そのデータを処理します。そして、各フィールドの記述に従って、ワークスペースのディメンションおよび変数にそのデータを格納します。
RECNO ファンクション	読み込みのために開かれているファイルの現在のレコード番号を戻します。

ファイル読み込みプログラムの記述

ファイルを読み込む際、各フィールドのデータの書式を個々に変更できます。また、ワークスペース・オブジェクトに割り当てる前に、DML ファンクションを使用して情報を処理することができます。通常、ファイルの読み込みでは次の手順を実行します。

1. データファイルを開きます。
2. ファイルのデータを一度に1レコード（1行）ずつ読み込みます。
3. データを処理し、1つ以上のワークスペース・オブジェクトに割り当てます。
4. ファイルを閉じます。

FILEREAD コマンドと FILEVIEW コマンドは同じ属性を持ち、データに対して同じ処理を行うことができます。ただし、次の点が大きく異なります。

- FILEREAD コマンドは、ファイル内のすべてのレコードに対して自動的に繰り返し実行され、レコードを自動的に処理します。ファイル内の読み込み対象のレコードがすべて同じである場合は、FILEREAD を使用します。FILEREAD は FILEVIEW よりも簡単に使用でき、処理も高速です。

ほとんどのファイルは FILEREAD で処理できるので、この章の例ではこのコマンドを使用します。

- FILEVIEW コマンドは、レコードを一度に1つずつ処理します。FILEVIEW は、もう1つのファイル読み込みコマンドよりも強力です。このコマンドは、FILEREAD で処理可能なすべてのファイルに加えて、別のタイプのレコードも処理できます。

例 11-1 は、OLAP DML でファイル読み込みプログラムを開発するためのテンプレートです。ディメンション・メンバーの読み込み方針については、11-13 ページの「[ディメンション・メンバーの表からのフェッチ](#)」を参照してください。

例 11-1 フラット・ファイルを読み込むためのテンプレート

```
VARIABLE funit INTEGER           "Define local variable for file handle
TRAP ON CLEANUP                 "Divert processing on error to CLEANUP label
funit = FILEOPEN('directory/datafile' READ) "Open the file

"Read the file with FILEREAD
FILEREAD funit
.
.
.
CLEANUP:                         "Cleanup label
IF funit NE na                   "Close the file
  THEN FILECLOSE funit
```

ワークスペース・オブジェクトへのフィールドのマッピング

FILEREAD コマンドは、フィールドをワークスペース・オブジェクトにマップします。次のいずれかの方法でレコードが構成されているファイルをソース・ファイルにすることができます。

- 列にデータを格納する書式指定ファイル。各フィールドは、開始位置および幅で定義されています。
- テキストまたは数値の文字列を格納する構造化 PRN ファイル。テキスト・フィールドは引用符で囲まれています。数値フィールドには、数値の他にピリオド (.) を格納できます。ただし、それ以外の文字（空白およびカンマを含む）はフィールドの区切りになります。
- 特殊文字（デリミタ）を使用してレコードのフィールドを区切った CSV（カンマ区切り）ファイル。

アナリティック・ワークスペースにおけるデータのターゲットは、ディメンション、リレーションまたは変数です。ディメンションは、新しいメンバーを追加してメンテナンスしたり、入力データを既存のディメンション・メンバーと単に揃えて配置するために使用できます。通常、スタンダード・フォームのアナリティック・ワークスペースでは、変数は属性またはメジャーです。

書式指定ファイルの読み込み

書式指定ファイルのデータをマッピングするための FILEREAD の基本構文は次のとおりです。

COLUMN *n* WIDTH *n* workspace_object

次の例は、データファイル内の 4 つのレコードを示しています。左から、channels 列、products 列、customers 列、time periods 列および units 列です。先頭の列（channels）の幅は 10 文字で、その他の列の幅は 11 文字です。

2	13	51	54	2
2	13	51	56	2
2	13	51	57	2
2	13	51	58	2

次の FILEREAD コマンドは、最終列のデータを UNITS_VARIABLE 変数に読み込み、残りの 4 つの列の値は既存のディメンション・メンバーに揃えて配置します。書式指定ファイルがデフォルトのレコード形式なので、RULED キーワードはオプションです。

```
FILEREAD funit RULED -
  COLUMN 1 WIDTH 10 channel -
  COLUMN 11 WIDTH 11 product -
  COLUMN 22 WIDTH 11 customer -
  COLUMN 33 WIDTH 11 time -
  COLUMN 44 WIDTH 11 units_variable
```


構造化 PRN ファイルの読み込み

構造化データをマッピングするための `FILEREAD` の基本構文は次のとおりです。

```
FIELD n workspace_object
```

「[書式指定ファイルの読み込み](#)」に示したものと同一データファイルは、次のコマンドで読み込むことができます。

```
FILEREAD funit STRUCTURED -  
  FIELD 1 channel -  
  FIELD 2 product -  
  FIELD 3 customer -  
  FIELD 4 time -  
  FIELD 5 units_variable
```

CSV ファイルの読み込み

CSV ファイルを読み込むための基本構文は、構造化 PRN ファイルの構文と同じです。

```
FIELD n workspace_object
```

次の例は、CSV ファイル内の 4 つのレコードを示しています。デリミタはカンマです。フィールドは「[書式指定ファイルの読み込み](#)」に示したデータファイルのものと同一で、channels、products、customers、time periods および units です。

```
2,13,51,54,2  
2,13,51,56,2  
2,13,51,57,2  
2,13,51,58,2
```

このファイルは、次のコマンドで読み込むことができます。カンマがデフォルトのデリミタなので、この場合、`DELIMITER` 句はオプションです。

```
FILEREAD funit CSV DELIMITER ',' -  
  FIELD 1 channel -  
  FIELD 2 product -  
  FIELD 3 customer -  
  FIELD 4 time -  
  FIELD 5 units_variable
```

メジャーを読み込むためのディメンション・ステータスの設定

データ値を変数に読み込む際は、必ず各ディメンションのステータスを設定する必要があります。入力レコードには通常、各ディメンションに対応したフィールドがあります。レコードがアナリティック・ワークスペースで処理されるとき、ディメンションは一時的にこれらの値に制限されます。これは、変数またはリレーションをターゲットとするデータが正しいセルに格納されるようにするためです。ただし、レコードで 1 つ以上のディメンションが省

略されている場合は、ファイルを読み込む前にこれらのディメンションを手動で設定する必要があります。

たとえば、ファイルに 2003 年 8 月の Direct Sales チャンネルのデータしか格納されておらず、チャンネルまたは時間を指定するフィールドがない場合は、ファイルを読み込む前に、プログラムで CHANNEL ディメンションおよび TIME ディメンションを制限する必要があります。制限しない場合、データは、これらのディメンションの最初のメンバー (All Channels および Jan-98) に揃えて配置されます。

データ・ロードの最適化

アナリティック・ワークスペースの変数が、最初に変化するディメンションおよび最後に変化するディメンションで定義されており、それらがソース・データファイル内のレコードの順序に一致する場合、データのロード時間が最短になります。ソース・データファイル内のレコードの順序を変更することが可能であれば、アナリティック・ワークスペース内の変数に一致するデータファイルを作成できます。それができない場合は、アナリティック・ワークスペース内の変数のディメンション順序を定義する際に、ロード向けに最適化するか、問合せ向けに最適化するかを選択する必要があります。

たとえば、データファイルにレコードが次の順序で格納されているとします。

- 1 目目のチャンネルの全レコード、次に 2 目目のチャンネルの全レコード (以下同様に続く)
- 1 目目のチャンネルの全製品、次に 2 目目のチャンネルの全製品 (以下同様に続く)
- 1 目目の製品の全顧客、次に 2 目目の製品の全顧客 (以下同様に続く)
- 1 目目の顧客の全期間、次に 2 目目の顧客の全期間 (以下同様に続く)

ワークスペースの変数定義においては、ディメンションの順序によってデータの格納方法が決まります。最初に変化するディメンションが先頭に、最後に変化するディメンションが末尾にリストされます。

このサンプル・ファイルの場合、TIME が最初に変化するディメンション、CHANNEL が最後に変化するディメンションとしてターゲット変数が定義されている場合に、データ・ロードが最短時間で行われます。したがって、ディメンションの順序は、TIME PRODUCT CUSTOMER CHANNEL となります。コンポジット・ディメンションの場合、定義は次のようになります。

```
DEFINE UNITS_VARIABLE VARIABLE DECIMAL <TIME UNITS_CUBE_COMPOSITE <CUSTOMER PRODUCT CHANNEL>>
```

TIME ディメンションを、コンポジットの外部で最初に変化するディメンションにすることで、時間ベースの分析における実行時のパフォーマンスも向上します。その理由は、期間が 1 つにまとめられるからです。これは、ワークスペースの変数が問合せ向けに最適化されていて、データファイルが最短時間でロードするために正しくソートされているという最良のシナリオです。

ただし、期間ごとに別々のデータファイルがある場合、ロードでは、TIME が最後に変化するディメンションになります。この場合、問合せを最適化するディメンション順序と、データ・ロードを最適化するディメンション順序が一致しくなくなります。このような環境において、どのディメンション順序が最適かを判断する必要があります。

データをロードするバッチ・ウィンドウが短い場合は、次のように TIME を最後のディメンションとして変数を定義することで、データ・ロード向けに最適化することが必要な場合があります。

```
DEFINE UNITS_VARIABLE VARIABLE DECIMAL <UNITS_CUBE_COMPOSITE <CUSTOMER PRODUCT
CHANNEL> TIME>
```

ディメンション・メンバーの読み込みとメンテナンス

通常、データファイルのレコードには、ディメンション値を格納するためのフィールドがあります。このフィールドは、データ値を格納するべきセルを識別します。データファイル内のすべてのディメンション値がすでにアナリティック・ワークスペースに存在している場合は、ディメンション・フィールドの記述でデフォルト属性の MATCH を使用できます。MATCH を使用すると、すでにアナリティック・ワークスペースに存在しているディメンション値のみが受け入れられるようになります。

入力値が一致しない場合、コマンドはエラーを発生させます。ファイル読み込みプログラムでエラーを処理するには、当該のレコードをスキップして処理を続行するか、処理を停止し、データファイルの妥当性をチェックするようユーザーに通知します。新しいディメンション・メンバーによってエラーが引き起こされたのか、または別のタイプのエラーなのかは、エラー発生時の ERRORNAME の一時値に基づいて判断します。

例 11-2 は、データファイル内のディメンション値がアナリティック・ワークスペース内のディメンション値と一致しなかった場合でも処理を続行できるようにするエラー処理のテンプレートです。

データファイルのディメンション値がすべて新しいものである場合、または新しいディメンション値と既存のディメンション値が混在している場合は、次のようにフィールドの記述で APPEND 属性を使用することで、新しい値および関連するすべてのデータをアナリティック・ワークスペースに追加できます。

```
FILEREAD funit -
    COLUMN n APPEND WIDTH n dimension
```

例 11-2 新しいディメンション・メンバーを持つレコードをスキップするテンプレート

```
VARIABLE funit INTEGER           "Define local variable for file handle
TRAP ON oops                     "Divert processing on error to oops label
funit = FILEOPEN('directory/datafile' READ) "Open the file

next:                             "Resume processing label
FILEREAD funit                    "Read the file with FILEREAD
.
```

```

      .
      .

WHILE FILENEXT(funit)                                "Or read it with FILEVIEW
DO
  FILEVIEW funit...
DOEND

FILECLOSE funit                                       "Close the file
RETURN                                                "End normal processing
oops:                                                 "Error label
IF funit NE na AND ERRORNAME NE 'ATTN'
  THEN DO
    TRAP ON oops
    GOTO next                                         "Resume processing at next label
  DOEND
IF funit NE na                                       "Close the file on error
  THEN FILECLOSE funit

```

入力値の変換

FILEREAD コマンドでは、アナリティック・ワークスペースへの読み込み時に値を変換できます。

基本的な変換

値の前後に単純に文字を追加するには、LSET 句または RSET 句を使用します。たとえば、入力の間が月のみの場合（JAN、FEB、MAR など）は、次のようにすると、値を TIME ディメンションに格納する前に年を追加できます。

```

FILEREAD funit -
  COLUMN 1 WIDTH 15 RSET '-04' time

```

その他の変換には、FILEVIEW コマンドや OLAP DML の任意のデータ操作ファンクションを使用できます。これらの操作の対象は、現在のフィールドの値を表す VALUE キーワードです。次の例では、入力値が大文字に変換されます。

```

FILEREAD funit -
  COLUMN 1 WIDTH 15 time = UPCASE(VALUE)

```

リレーションを使用したディメンション値の調整

既存のディメンション値に一致させる必要があり、単純な変換では一致させることができない場合は、2つの値のセットを相関させるリレーションをアナリティック・ワークスペースに作成できます。次の手順を実行します。

1. 入力ディメンション値のためのディメンションを新規作成します。

ディメンションを定義するには、Analytic Workspace Manager を使用するか、または OLAP Worksheet で次のようなコマンドを発行します。

```
DEFINE new_dimension DIMENSION TEXT
```

2. ファイルからディメンション値を読み込みます。
3. 2つのディメンション間にリレーションを作成します。

リレーションを定義するには、Analytic Workspace Manager を使用するか、または OLAP Worksheet で次のようなコマンドを発行します。

```
DEFINE relation RELATION dimension <new_dimension>
```

4. データを揃えるためのリレーションを使用して、ファイルからデータを読み込みます。
次の構文を使用します。

```
FILEREAD funit CSV -  
FIELD 1 dimension = relation(new_dimension VALUE)
```

たとえば、Product ディメンションでディメンション・メンバーとして SKU（最小在庫管理単位）を使用していて、バーコードを使用するデータを追加したい場合は、バーコード用のディメンションと、バーコードと Product ディメンションの SKU 間のリレーションを作成します。リレーションは、バーコードと SKU を関連させるファイルから移入できます。

リレーショナル表からのデータのフェッチ

OLAP DML の SQL コマンドを使用すると、OLAP DML プログラムに SQL 文を埋め込むことができます。

```
SQL sql_statement
```

OLAP DML の引数となる SQL 文を記述する場合、引用符が必要な箇所には、一重引用符 (') を使用する必要があります。OLAP DML では、二重引用符 (") はコメントの始まりを示します。

OLAP DML の SQL サポート

OLAP DML の SQL コマンドでは、Oracle でサポートされているほとんどの SQL 文を使用できます。SELECT コマンドを使用すると、リレーショナル表のデータをアナリティック・ワークスペース・オブジェクトにコピーできます。INSERT コマンドを使用すると、アナリティック・ワークスペース・オブジェクトのデータをリレーショナル表にコピーできます。

次の Oracle SQL 拡張もサポートされています。

- カーソルに関連付けられているデータを更新または削除するための、カーソル宣言の SELECT 文における FOR UPDATE 句

- 対話的に表を変更するための、UPDATE 文および DELETE 文における WHERE CURRENT OF カーソル句
- ストアド・プロシージャおよびトリガー

SQL コマンドの引数として指定された COMMIT および ROLLBACK は無視されます。変更をコミットするには、OLAP DML の UPDATE コマンドおよび COMMIT コマンドを発行します。OLAP DML を使用して変更をロールバックすることはできません。

ほとんどの SQL コマンドは直接 SQL コマンド・プロセッサに送られますが、少数の一部コマンドは先に OLAP DML で処理されます。それらの構文は、標準 SQL と多少異なる場合があります。

表 11-2 に、埋込み SQL をサポートする OLAP DML コマンドを示します

表 11-2 埋込み SQL をサポートする OLAP DML コマンド

文	説明
SQL コマンド	SQL コマンドをデータベースの SQL コマンド・プロセッサに渡します。
SQLBLOCKMAX オプション	表から一度に取得するレコードの最大数を制御します。
SQLCODE オプション	最後に行われた SQL 操作の後でデータベースから戻された値を保持します。
SQLERRM オプション	SQLCODE がゼロ以外の値を保持する場合に、エラー・メッセージを格納します。
SQLMESSAGES オプション	エラー・メッセージを現在の出力ファイルに送信するかどうかを制御します。

プロセス：リレーショナル表からアナリティック・ワークスペース・オブジェクトへのデータのコピー

OLAP DML を使用してリレーショナル表からスタンダード・フォームのアナリティック・ワークスペースを移入するには、次の手順を実行します。

1. リレーショナル表のデータを保持するアナリティック・ワークスペース・オブジェクトを定義します。

11-2 ページの「[スタンダード・フォームのアナリティック・ワークスペースを手動で作成する方法](#)」の手順を実行します。次に、アナリティック・ワークスペースを参照して、移入する必要があるオブジェクトを指定します。
2. ディメンションごとに OLAP DML プログラムを記述します。プログラムをコンパイルおよび実行します。

「[ディメンション・メンバーの表からのフェッチ](#)」の説明を参照してください。

3. キューブごとに OLAP DML プログラムを記述します。プログラムをコンパイルおよび実行します。

11-15 ページの「[表からのメジャーのフェッチ](#)」の説明を参照してください。

ディメンション・メンバーの表からのフェッチ

ディメンション・メンバーのフェッチには複数の方針があります。最良の方針は、最初にディメンション・メンバーのみをフェッチし、別の手順で属性をフェッチすることです。Time のメンバーについては、レベルを一度に 1 つずつフェッチし、ソース表への経路をレベルごとに別々にするのが最良の方針です。これにより、Time のメンバーを正しい順序でフェッチできるので、後でメンバーをソートせずに済みます。

しかし、最も単純な方法およびここで説明する方法では、すべてのレベルのディメンション・メンバー、および階層をサポートするすべてのオブジェクトを同時に移入します。この方法を実行する前に、6-9 ページの「[セグメント・サイズの設定](#)」で説明した SEGWIDTH が正しく設定されていることを確認してください。

例 11-3 は、1 つの経路でディメンションをフェッチするために使用できるテンプレートです。プログラムでは、次の処理を行います。

- レベル列を、その属性列とともに一度に 1 つずつ読み込みます。最上位のレベル（最大集計レベル）から開始し、ベース・レベルまで読み込みます。この構文でサポートされる階層は 1 つです。複数の階層を処理するための FILEREAD による同等の構文については、11-32 ページの例 11-18 を参照してください。

親リレーションも同時に移入されるので、子を追加する前に親をワークスペース・ディメンションに追加しておく必要があります。親を先に追加しないと、エラーが発生します。これは、リレーションは特定のディメンションのメンバーのみを有効な値として受け入れるためです。すべてのディメンション・メンバーを先にロードする場合、これは問題になりません。

- ブール変数 *member_inhier* を手動で移入します。構文の *n* は、1（Yes の場合）または 0（No の場合）になります。
- *member_levelrel* リレーションに、列の適切なレベル名を手動で移入します。レベル名は、*levellist* ディメンションのメンバーと完全に一致している必要があります。
- *member_parentrel* リレーションに、適切な列から親ディメンション・メンバーを移入します。
- エラー処理のためのコマンドを記述します。他の例では、コードを読みやすくするためにこれらのコマンドは省略されています。

例 11-3 ディメンション・メンバーをフェッチするためのテンプレート

```
SQLMESSAGES=YES          " Display error messages on the screen
TRAP ON CLEANUP          " Go to the CLEANUP label if an error occurs
SQL DECLARE cursor CURSOR FOR SELECT -
```

```
top_level, n, 'levelname', parent_level, attribute, attribute,...-
.
.
.
base_level, n, 'levelname', attribute, attribute,... -
FROM table -
WHERE where_clause

" Signal an error if the command failed
IF SQLCODE NE 0
    THEN SIGNAL declerr 'Define cursor failed'

" Open the cursor
SQL OPEN cursor
IF SQLCODE NE 0
    THEN SIGNAL openerr 'Open cursor failed'

" Fetch the data
SQL IMPORT cursor INTO -
    :APPEND dimension, :inhier, :levelrel, :parentrel, :attribute, :attribute, ...

IF SQLCODE NE 0 AND SQLCODE NE 100
    THEN SIGNAL geterr 'Fetch failed'

" Save these changes
UPDATE
COMMIT

CLEANUP:
SQL CLOSE cursor
SQL CLEANUP
```

ディメンション・メンバーのソート

ソース表から1つの経路ですべてのレベルのディメンション・メンバーをフェッチすると、ターゲットのワークスペース・ディメンションでは、すべてのレベルのディメンション・メンバーが混在することになります。必要に応じて、レベルごとにメンバーをソートすることはできますが、ほとんどのディメンションでは、順序が処理に影響を与えることはありません。

ただし、**Time** ディメンションで時系列分析をサポートするためには、**Time** ディメンションのメンバーをレベル内で時系列順に並べることが非常に重要になります。**LEAD**、**LAG**、**MOVINGAVERAGE** などのファンクションは、計算の中で**Time** のメンバーの相対位置を使用します。たとえば、**LAG** は、現在のメンバーよりも前にある、指定された数の値であるディメンション・メンバーを戻します。期間が時系列順になっていない場合、戻り値は意味をなしません。

ディメンションを後でソートするのではなく、正しい順序でディメンションをロードした場合の方が、アナリティック・ワークスペースのパフォーマンスは向上します。

例 11-4 は、Time ディメンションをソートするための OLAP DML プログラムのテンプレートです。このプログラムでは、次の処理を行います。

- ソート済の値を保持するための値セットを定義します。
- Time ディメンションを、レベル別、最終日別の順にレベル内でソートします。
- ソート済の値を値セットに格納します。
- Time ディメンションのメンバーを並べ替えます。

例 11-4 Time ディメンションをソートするためのテンプレート

```
IF NOT EXISTS('valueset')
  THEN DEFINE valueset VALUESET time_dim
LIMIT time_dim TO ALL
"Sort levels in descending order and time periods in ascending order
SORT time_dim D time_dim_LEVELREL A time_dim_END_DATE
LIMIT valueset TO time_dim
MAINTAIN time_dim MOVE VALUES(valueset) FIRST

"Save these changes
UPDATE
COMMIT
```

表からのメジャーのフェッチ

リレーショナル表からデータをフェッチするには、フェッチするデータを選択する SQL カーソルを定義し、作成済のアナリティック・ワークスペース・オブジェクトにデータを取得するというプログラムを記述します。

例 11-5 は、特定のキューブのすべてのメジャーをフェッチするために使用できるテンプレートです。このプログラムでは、次の処理を行います。

- 各ディメンションのメンバーを持つキー列を識別します。データが変数にフェッチされると、これらの値によってディメンションが適切なセルに制限されます。
- ターゲット変数と一致するようにソース・データを並べ替えます。

SELECT 文の ORDER BY 句は、変数のディメンション・リストと逆になります。Product が最初に変化するディメンション、Time が最後に変化するディメンションになるように変数が <Product Geography Time> でディメンション化されている場合、ORDER BY 句では、Time、Geography、Product の順に行をソートします。この順序によってデータ・ロードの速度が向上します。

- キー列の値と、ターゲットのワークスペース・ディメンションの既存メンバーとの一致を必須にします。

一致を必須にすることで、レベル、親子関係、属性などを持たない新しいディメンション・メンバーが作成されないようにします。

例 11-5 メジャーをフェッチするためのテンプレート

```
SQLMESSAGES=YES          " Display error messages on the screen
TRAP ON CLEANUP           " Go to the CLEANUP label if an error occurs
" Define a cursor for selecting data
SQL DECLARE cursor CURSOR FOR SELECT -
    key1, key2, key3, keyn -
    meas1, meas2, meas3, measn -
    FROM table
    ORDER BY slowest_dim, ..., fastest_dim
" Signal an error if the command failed
IF SQLCODE NE 0
    THEN SIGNAL declerr 'Define cursor failed'

" Open the cursor
SQL OPEN cursor
IF SQLCODE NE 0
    THEN SIGNAL openerr 'Open cursor failed'

" Fetch the data
SQL IMPORT cursor INTO :MATCH dim1, :MATCH dim2, -
    :MATCH dim3, :MATCH dimn, -
    :var1, :var2 :var3
IF SQLCODE NE 0 AND SQLCODE NE 100
    THEN SIGNAL geterr 'Fetch failed'

" Save these changes
UPDATE
COMMIT

CLEANUP:
SQL CLOSE cursor " Close the cursor
SQL CLEANUP      " Free resources
```

その他のメタデータ・オブジェクトの移入

アナリティック・ワークスペースの一部のメタデータ・オブジェクトは、データをロードした後でのみ移入できます。ここではスタンダード・フォームのアナリティック・ワークスペースを作成しているので、DBMS_AWM パッケージと同じ OLAP DML プログラムを使用できます。これらのプログラムは、SYS が所有する AWCREATE というアナリティック・ワークスペースに格納されています。次の OLAP DML コマンドでこのワークスペースをアタッチすることにより、これらのプログラムにアクセスできます。

```
AW ATTACH sys.awcreate
```

この項では、次の 2 つのプログラムについて説明します。

- `___POP.FMLYREL` は、`member_gid` 変数および `member_familyrel` リレーションを移入します。
- `___ORDER.HIERARCHIES` は、`default_order` 変数を移入します。

プログラム名の先頭には 3 つのアンダースコアが付きます。

___POP.FMLYREL の使用

`___POP.FMLYREL` プログラムは、データ・キューブのディメンションの `member_gid` 変数および `member_familyrel` 変数を移入します。`___POP.FMLYREL` は、ディメンションごとに実行する必要があります。`___POP.FMLYREL` をコールするには、次の構文を使用します。

```
CALL ___POP.FMLYREL(aw, aw!dim, aw!dim_HIERLIST, aw!dim_LEVELLIST, aw!dim_LEVELREL,
dim, aw!dim_PARENTREL, aw!dim_INHIER)
```

各項目の意味は次のとおりです。

`aw` はアナリティック・ワークスペースの名前です。

`dim` は `dimdef` ディメンションの名前です。

引数はすべてテキスト式なので、リテラル・テキストは一重引用符で囲む必要があります。引数はすべて大文字にします。

例については、11-36 ページの「[その他のスタンダード・フォームのメタデータ・オブジェクトの移入](#)」を参照してください。

___ORDR.HIERARCHIES の使用

`___ORDR.HIERARCHIES` プログラムは、データ・ディメンションの `default_order` 属性を移入します。このプログラムは、データ・キューブのディメンションごとに実行する必要があります。`___ORDR.HIERARCHIES` を実行するには、次の構文を使用します。

```
CALL ___ordr.hierarchies('aw!dim', 'aw!dim_HIERLIST', 'aw!dim_HIER_CREATEDBY',
'dim_PARENTREL', 'dim_ORDER', 'dim_INHIER')
```

各項目の意味は次のとおりです。

`aw` はアナリティック・ワークスペースの名前です。

`dim` は `dimdef` ディメンションの名前です。

引数はすべてテキスト式なので、リテラル・テキストは一重引用符で囲む必要があります。引数はすべて大文字にします。

例については、11-36 ページの「[その他のスタンダード・フォームのメタデータ・オブジェクトの移入](#)」を参照してください。

事歴：他のソースからの GLOBALX ワークスペースの作成

この例では、リレーショナル表およびフラット・ファイルからデータを取得してアナリティック・ワークスペースを作成する方法を示します。ここでは、これまでも扱ってきた Global データを使用します。Global スター・スキーマから直接アナリティック・ワークスペースを作成する場合は、[第 6 章](#)を参照してください。

基本的な手順は次のとおりです。

1. GLOBALX ユーザーおよびデフォルトの表領域を作成します。
2. GLOBALX にスター・スキーマを作成します。
3. すべてのディメンション、レベル、階層、属性およびメジャーを定義する、GLOBALX スター・スキーマの OLAP カタログ・メタデータを作成します。
4. GLOBALX_AW ユーザーおよびデフォルトの表領域を定義します。
5. スタンダード・フォームのアナリティック・ワークスペース GLOBALX を作成します。
6. コンポジットの再定義やセグメント・サイズの設定など、GLOBALX アナリティック・ワークスペースに変更を加えます。
7. OLAP DML の SQL コマンドを使用して、リレーショナル表から Price キューブを移入します。
8. OLAP DML のファイル読み込みコマンドを使用して、フラット・ファイルから Units キューブを移入します。
9. データを集計します。
10. GLOBALX アナリティック・ワークスペースを、OLAP API および BI Beans から使用できるようにします。

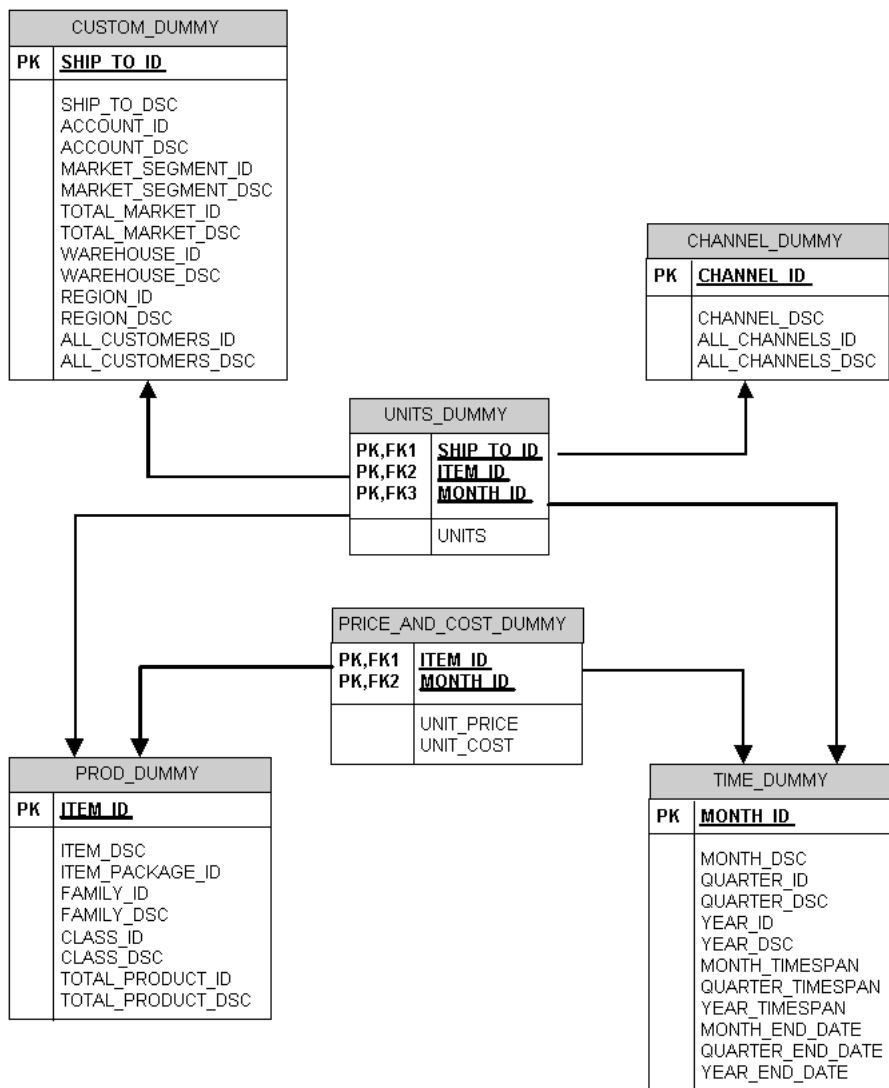
GLOBALX スター・スキーマの設計および実装

Global データはすでに GLOBAL スター・スキーマに格納されているので、GLOBALX では、Price キューブおよび Units キューブの設計を流用できます。両スター・スキーマの違いは、GLOBAL にはデータが移入されていて、GLOBALX には空の表が格納されているということです。

GLOBALX スキーマの図

[例 11-6](#) に、スキーマの関係図を示します。

例 11-6 GLOBALX スキーマの図



手順：GLOBALX サンプル・スキーマの作成

サンプルの GLOBALX スキーマを作成するには、次の手順を実行します。

1. GLOBALX ユーザーおよびデフォルトの表領域を作成します。サンプル・スクリプトを B-1 ページの「ユーザーおよび表領域の定義用の SQL スクリプト」に示します。
2. B-2 ページの「GLOBALX スター・スキーマ用の SQL スクリプト」に示す SQL スクリプトを作成します。
3. SQL*Plus または同様の SQL コマンド・プロセッサに GLOBALX ユーザーとしてログインします。
4. SQL の @ コマンドを使用してスクリプトを実行します。
5. エラーが発生することなくスクリプトが実行されたら、SQL の COMMIT 文を発行します。

GLOBALX スキーマの OLAP カタログ・メタデータの作成

GLOBALX スター・スキーマのメタデータは、Oracle Enterprise Manager の OLAP 管理ツール、Oracle Warehouse Builder の OLAP Bridge または CWM2 PL/SQL パッケージのうち、利用可能な任意の方法で生成できます。この例では、CWM2 パッケージを使用します。

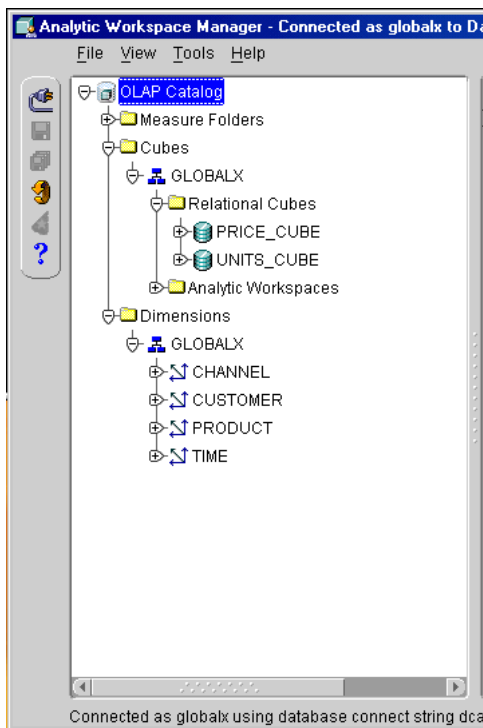
GLOBALX スキーマの OLAP カタログ・メタデータを作成するには、次の手順を実行します。

1. B-4 ページの「OLAP カタログ・メタデータ用の SQL スクリプト」に示す SQL スクリプトを作成します。
2. SQL*Plus または同様の SQL コマンド・プロセッサに GLOBALX ユーザーとしてログインします。
3. 次の SQL コマンドを発行して、メタデータ・バリデータによるレポートが完全に表示されるようにします。

```
SET LINESIZE 135
SET SERVEROUT ON
EXECUTE cwm2_olap_manager.set_echo_on
```

4. SQL の @ コマンドを使用して CWM2 スクリプトを実行します。
5. エラーが発生することなくスクリプトが実行されたら、SQL の COMMIT 文を発行します。
6. Analytic Workspace Manager でメタデータを確認します (図 11-1 を参照)。
 - a. Analytic Workspace Manager を開き、GLOBALX ユーザーとして接続します。
 - b. OLAP カタログ・ビューで、「Cubes」、「GLOBALX」および「Relational Cubes」の各フォルダを開きます。

図 11-1 Analytic Workspace Manager に表示された GLOBALX メタデータ



GLOBALX アナリティック・ワークスペースの作成

GLOBALX アナリティック・ワークスペースは、Analytic Workspace Manager のアナリティック・ワークスペース作成ウィザード、Oracle Warehouse Builder の OLAP Bridge または DBMS_AWM PL/SQL プロシージャのうち、利用可能な任意の方法を使用して、空の表および OLAP カタログ・メタデータから作成できます。この例では、ウィザードを使用して DBMS_AWM へのコールを含むスクリプトを生成し、このスクリプトに変更を加えます。

GLOBALX アナリティック・ワークスペースを作成するには、次の手順を実行します。

1. B-1 ページの「ユーザーおよび表領域の定義用の SQL スクリプト」に示すようなスクリプトを使用して、GLOBALX の表および表領域へのアクセス権を持つ GLOBALX_AW ユーザーを作成します。
2. Analytic Workspace Manager を開き、GLOBALX_AW ユーザーとしてデータベースにログインします。
3. 「Tools」メニューから「Create Analytic Workspace Using Wizard」を選択します。

4. ウィザードで次の設定を行います。

- 「Specify Analytic Workspace」 ページ： ワークスペース名として「GLOBALX」と入力し、スキーマとして「GLOBALX_AW」を選択します。
- 「Select Cubes」 ページ： GLOBALX リレーショナル・スキーマ内のすべてのキューブを選択します。
- 「Choose Data Loading Options」 ページ： 「**Build analytic workspace and load dimensions and facts**」を選択します。「**Generate unique keys**」ボックスを選択解除します。

表の中にはロードするデータは格納されていませんが、このオプションを選択すると、他のオプションよりも多くのカタログがアナリティック・ワークスペースに移入されます。データがないことが原因でアナリティック・ワークスペースの作成に失敗することはありません。

- 「Choose Advanced Storage and Naming Options」 ページ： 「**Display the pages for setting the advanced storage options**」を選択します。「**Prefix measure names with cube names**」ボックスを選択解除します。

使用可能なデータが存在しないので、ツールは、ディメンションの正しい順序および適切なセグメント・サイズを判断できません。この情報はユーザーが指定する必要があります。指定しない場合、アナリティック・ワークスペースのパフォーマンスが低下します。

- 「Create Composite Dimension」以降のページ： UNITS_CUBE_COMPOSITE という名前で Units キューブのコンポジットを作成します。ディメンションの順序は、CUSTOMER PRODUCT CHANNEL とします。TIME は、コンポジットから省きます。
- 「Specify Segment Width and Dimension Order」 ページ： Units キューブに対し、ディメンションを次のように指定します。

```
TIME                                85
<CUSTOMER PRODUCT CHANNEL>        1000000
```

5. 新しいアナリティック・ワークスペースを保存します。
6. OLAP Worksheet で GLOBALX アナリティック・ワークスペースを開き、次の変更を加えます。

- a. UNIT_COST_VARIABLE および UNIT_PRICE_VARIABLE を削除して、<TIME PRODUCT> でディメンション化されるようにこれらを再定義します。

稠密度 80% のこれらの 2 次元メジャーでは、コンポジットがない方がパフォーマンスが向上します。コンポジットの定義については、6-12 ページの「[GLOBAL のスパー特性の調査](#)」を参照してください。6-13 ページの「[オブジェクト定義の手動による変更](#)」の手順を実行します。

- b. PRICE_CUBE_COMPOSITE を削除します。

- c. 次のようなコマンドで、どちらかの変数にセグメント・サイズを設定します。

```
CHGDFN unit_price_variable SEGWIDTH 85 50
```

これらの設定により、85 の期間と 50 の製品用に、連続したディスク領域が確保されます。1 つのコマンドで、同一キューブ内のすべてのメジャーのセグメント・サイズが変更されます。

- d. 次のコマンドで、ディメンション属性のセグメント・サイズを設定します。

```
CHGDFN time_end_date SEGWIDTH 85 1
CHGDFN time_long_description SEGWIDTH 85 1 1
CHGDFN customer_long_description SEGWIDTH 80 2 1
CHGDFN product_long_description SEGWIDTH 50 1 1
CHGDFN channel_long_description SEGWIDTH 3 1 1
```

- e. UPDATE コマンドおよび COMMIT コマンドを発行して、これらの変更を保存します。

リレーショナル表からの Price キューブのフェッチ

Price キューブには、PRODUCT および TIME という 2 つのディメンションを持つ、UNIT_COST および UNIT_PRICE という 2 つのメジャーがあります。この例では、GLOBAL スター・スキーマから手動でデータをロードします。ただし、ここで説明している方法で、あらゆる形式のリレーショナル表からデータをロードすることが可能です。

GLOBALX アナリティック・ワークスペースにおける Price キューブの移入には、次の手順を実行します。

1. Analytic Workspace Manager を開き、GLOBALX_AW ユーザーとしてデータベースに接続します。
2. オブジェクト・ビューで、GLOBALX アナリティック・ワークスペースを読み込み / 書き込みモードで開きます。
3. PRODUCT ディメンションおよび TIME ディメンションのメンバーをフェッチするための OLAP DML プログラムを作成します。

プログラムの作成とコンパイルは、オブジェクト・ビューまたは OLAP Worksheet で行うことができます。プログラムの実行と、そのプログラムによって移入されたオブジェクトの内容の表示は、OLAP Worksheet でのみ行うことができます。

4. OLAP Worksheet を開き、CALL コマンドを使用してプログラムを実行します。

```
CALL program_name
```

5. エラーが発生することなくプログラムが実行されたら、ターゲットのワークスペース・オブジェクトの内容をチェックして、移入が正しく行われたかどうかを確認します。

6. UPDATE コマンドおよび COMMIT コマンドを発行して、ロードされたデータを保存します。
7. Price キューブ用のデータ・ロード・プログラムを作成および実行します。
8. エラーが発生することなくプログラムが実行されたら、ターゲット変数内のデータをチェックします。
9. UPDATE コマンドおよび COMMIT コマンドを発行して、ロードされたデータを保存します。

GLOBAL.PRODUCT_DIM からの Products のロード

例 11-7 の GETPROD プログラムは、PRODUCT ディメンション、*member_parentrel* リレーションおよび *long_description* 変数にデータをフェッチします。親値は、その子よりも先に PRODUCT ディメンションに追加する必要があることに注意してください。このようにしないと、親リレーションを移入する際にエラーが発生します。このため、SELECT 文では、集計の最高レベルから最低レベルへの順でレベル列を指定しています。

ブール変数 *member_inhier* には、1 (true の場合) または 0 (false の場合) が移入されます。また、*member_levelrel* リレーションには、*levellist* ディメンションの値と一致するテキスト値が移入されます。

サンプル・プログラムを定義し、次のコマンドでプログラムを実行します。

```
CALL getprod
```

例 11-7 GLOBAL.PRODUCT_DIM から Products をロードする OLAP DML プログラム

```
DEFINE GETPROD PROGRAM
PROGRAM
TRAP ON CLEANUP
" Define cursor c1
SQL DECLARE c1 CURSOR FOR SELECT -
    total_product_id, 1, 'TOTAL_PRODUCT', total_product_dsc, -
    class_id, 1, 'CLASS', total_product_id, class_dsc, -
    family_id, 1, 'FAMILY', class_id, family_dsc, -
    item_id, 1, 'ITEM', family_id, item_dsc, item_package_id -
FROM global.product_dim

" Open the cursor
SQL OPEN c1

" Fetch the data
SQL FETCH c1 LOOP INTO -
:APPEND product, :product_inhier, :product_levelrel, :product_long_description, -
:APPEND product, :product_inhier, :product_levelrel, :product_parentrel, -
    :product_long_description, -
:APPEND product, :product_inhier, :product_levelrel, :product_parentrel,-
```

```
:product_long_description, -
:APPEND product, :product_inhier, :product_levelrel, :product_parentrel, -
:product_long_description, :product_package

" Save these changes
UPDATE
COMMIT

CLEANUP:
SQL CLOSE c1
SQL CLEANUP
END
```

例 11-8 では、ロードが正常に行われたかどうかを確認するためにデータを選択しています。

例 11-8 PRODUCT ディメンションおよび属性の表示

```
LIMIT product TO product_levelrel EQ 'ITEM'
LIMIT product KEEP FIRST 2
LIMIT product ADD ANCESTORS USING product_parentrel
REPORT W 8 DOWN product W 16 <product_long_description product_levelrel>
      W 10 <product_parentrel product_inhier>
```

ALL_LANGUAGES: AMERICAN_AMERICA

-----PRODUCT_HIERLIST-----				
-----PRODUCT_ROLLUP-----				
PRODUCT	PRODUCT_LONG_DES CRPTION	PRODUCT_LEVELREL	PRODUCT_PA RENTREL	PRODUCT_IN HIER
13	Envoy Standard	ITEM	4	yes
14	Envoy Executive	ITEM	4	yes
4	Portable PCs	FAMILY	2	yes
2	Hardware	CLASS	1	yes
1	Total Product	TOTAL_PRODUCT	NA	yes

GLOBAL.TIME_DIM からの Time のロード

TIME のメンバーをフェッチするプログラム (例 11-9) は、前述の PRODUCT のメンバーをフェッチするプログラムとよく似ています。唯一の違いは、期間属性および最終日属性が追加されていることです。

ただし、TIME のメンバーは、時系列分析ファンクションをサポートするために、レベル内で時系列順にソートする必要があります。各行には、各レベルのディメンション・メンバーが格納されています。そのため、TIME ディメンションには、完全に混在した状態でレベルが移入されます。例 11-10 は、TIME ディメンションをソートするプログラムです。このプログラムは、SORT コマンドを使用して TIME ディメンションの現在の一時ステータスを並

べ替え、この順序を値セットに保存します。そして、MAINTAIN コマンドを値セットに対して繰り返し実行し、値を永続的に並べ替えます。

サンプル・プログラムを定義し、次のコマンドでプログラムを実行します。

```
CALL gettime
CALL timesort
```

例 11-9 GLOBAL.TIME_DIM から Time をロードする OLAP DML プログラム

```
DEFINE GETTIME PROGRAM
PROGRAM
TRAP ON CLEANUP
SQL DECLARE c1 CURSOR FOR SELECT -
    year_id, 1, 'YEAR', year_dsc, year_timespan, year_end_date, -
    quarter_id, 1, 'QUARTER', year_id, quarter_dsc, quarter_timespan, quarter_end_date, -
    month_id, 1, 'MONTH', quarter_id, month_dsc, month_timespan, month_end_date -
FROM global.time_dim -
ORDER BY month_end_date

" Open the cursor
SQL OPEN c1

" Fetch the data
SQL FETCH c1 LOOP INTO -
:APPEND time, :time_inhier, :time_levelrel, :time_long_description, -
    :time_time_span, :time_end_date,-
:APPEND time, :time_inhier, :time_levelrel, :time_parentrel, -
    :time_long_description, :time_time_span, :time_end_date,-
:APPEND time, :time_inhier, :time_levelrel, :time_parentrel, -
    :time_long_description, :time_time_span, :time_end_date

" Save these changes
UPDATE
COMMIT

CLEANUP:
SQL CLOSE c1
SQL CLEANUP
END
```

例 11-10 TIME ディメンションのメンバーをソートする OLAP DML プログラム

```
DEFINE TIMESORT PROGRAM
PROGRAM
" Create a valueset to hold the sorted values
IF NOT EXISTS('timeset')
    THEN DEFINE timeset VALUESET time
```

```
LIMIT time TO ALL
" Sort by descending levels and ascending end-dates
SORT time D time_LEVELREL A time_end_date
" Save sorted values in the valueset
LIMIT timeset TO time
" Reorder the dimension members
MAINTAIN time MOVE VALUES(timeset) FIRST
END
```

TIME ディメンションには非常に多くのメンバーが存在するため、すべてのメンバーを表示することはできません。ただし、PRODUCT の場合のように祖先ごとにメンバーを選択することで、一時的にディメンションを並べ替えることができます。その結果から、オブジェクトが正しく移入されたかどうかを確認することはできますが、メンバーが正しくソートされているかどうかは必ずしも確認できません。例 11-11 では、元の順序を変更しない LIMIT コマンドを使用しています。レポートから、ソートが正しく行われていることがわかります。

例 11-11 TIME ディメンションおよび属性の表示

```
LIMIT time TO FIRST 10
LIMIT time ADD LAST 3
REPORT W 5 DOWN time W 8 <time_long_description time_parentrel time_levelrel
time_inhier time_end_date time_time_span>
```

ALL_LANGUAGES: AMERICAN_AMERICA						
-----TIME_HIERLIST-----						
-----CALENDAR-----						
TIME_LON						
G_DESCRI TIME_PAR TIME_LEV TIME_INH TIME_END TIME_TIM						
TIME	PTION	ENTREL	ELREL	IER	_DATE	E_SPAN

1	1998	NA	YEAR	yes	31DEC98	365.00
2	1999	NA	YEAR	yes	31DEC99	365.00
3	2000	NA	YEAR	yes	31DEC00	366.00
4	2001	NA	YEAR	yes	31DEC01	365.00
85	2002	NA	YEAR	yes	31DEC02	365.00
102	2003	NA	YEAR	yes	31DEC03	365.00
119	2004	NA	YEAR	yes	31DEC04	366.00
5	Q1-98	1	QUARTER	yes	31MAR98	90.00
6	Q2-98	1	QUARTER	yes	30JUN98	91.00
7	Q3-98	1	QUARTER	yes	30SEP98	92.00
106	Apr-04	116	MONTH	yes	30APR04	30.00
107	May-04	116	MONTH	yes	31MAY04	31.00
108	Jun-04	116	MONTH	yes	30JUN04	30.00

PRICE_AND_COST_HISTORY_FACT からの PRICE キューブのロード

例 11-12 は、UNIT_PRICE_VARIABLE および UNIT_COST_VARIABLE にデータをフェッチするプログラムです。データは、論理メジャーと同じ名前の、*measuredef* 式ではなく、変数にロードする必要があることに注意してください。これらの変数の定義は次のとおりです。

```
DEFINE UNIT_PRICE_VARIABLE VARIABLE DECIMAL <TIME PRODUCT>
LD IMPLEMENTATION Variable for UNIT_PRICE Measure
```

```
DEFINE UNIT_COST_VARIABLE VARIABLE DECIMAL <TIME PRODUCT>
LD IMPLEMENTATION Variable for UNIT_COST Measure
```

DECLARE CURSOR SELECT 文の ORDER BY 句は、PRODUCT (ITEM_ID) が後で変化するディメンション、TIME (MONTH_ID) が先に変化するディメンションとなるように行をソートします。この編成は、その定義で示した、ワークスペース変数に値が格納される順序に対応しています。このようにソートすることで、データが最短時間でロードされるようになります。

ディメンション・メンバーはすべて、すでにアナリティック・ワークスペースに存在する必要があります。ディメンション・メンバーの中に一致するものが見つからない値がある場合、プログラムは正常に実行されず、エラーが発生します。

サンプル・プログラムを定義し、次のコマンドでプログラムを実行します。

```
CALL getpricecube
```

例 11-12 PRICE_AND_COST_HISTORY_FACT から PRICE キューブをロードする OLAP DML プログラム

```
DEFINE GETPRICECUBE PROGRAM
PROGRAM
" Define a cursor for selecting data
SQL DECLARE c1 CURSOR FOR SELECT -
    item_id, month_id, unit_price, unit_cost -
FROM global.price_and_cost_history_fact -
ORDER BY item_id, month_id

" Open the cursor
SQL OPEN c1

" Fetch the data
SQL FETCH c1 LOOP INTO :MATCH product, :MATCH time, -
    :unit_price_variable, :unit_cost_variable

" Save these changes
UPDATE
COMMIT
SQL CLOSE c1 " Close the cursor
SQL CLEANUP
END
```

ほとんどのメジャーと異なり、Price キューブのメジャーは稠密なので、データは簡単にチェックできます。例 11-13 の LIMIT コマンドは、PRODUCT 階層および TIME 階層のすべてのレベルのメンバーを選択します。データは最低レベルにのみ存在するので、その他のレベルはオンデマンドで計算されます。基になる変数ではなく、*measuredef* 式が示されていることに注意してください。

変数に値があるかどうかを簡単にチェックするには、次の ANY ファンクションを使用します。

```
SHOW ANY(variable NE NA)
```

次に例を示します。

```
SHOW ANY(unit_price_variable NE NA)
```

戻り値 YES は、1 つ以上のセルにデータがあることを示し、NO は、すべてのセルが空であることを示します。

例 11-13 PRICE_CUBE のデータ・ロードの検証

```
LIMIT time TO '44' '45'  
LIMIT time ADD ANCESTORS USING time_parentrel  
LIMIT product TO '13'  
LIMIT product ADD ANCESTORS USING product_parentrel  
REPORT unit_price unit_cost
```

TIME	-----PRODUCT-----							
	-----13-----		-----4-----		-----2-----		-----1-----	
	UNIT_PRICE	UNIT_COST	UNIT_PRICE	UNIT_COST	UNIT_PRICE	UNIT_COST	UNIT_PRICE	UNIT_COST
44	3,008.91	2,862.51	9,483.71	8,944.35	19,163.02	18,071.18	19,960.72	18,714.88
45	3,142.99	2,926.79	9,590.01	9,024.35	18,969.89	17,924.55	19,735.20	18,542.48
13	9,152.01	8,655.17	28,452.92	26,883.70	57,229.23	53,982.32	59,577.63	55,873.16
3	34,314.40	32,681.09	111,333.24	105,481.00	224,713.71	211,680.12	234,516.47	219,574.10

フラット・ファイルからの Units キューブのロード

Units キューブには、1 つのメジャー (UNITS) と 4 つのディメンション (TIME、CUSTOMER、PRODUCT および CHANNEL) があります。TIME ディメンションと PRODUCT ディメンションは、11-23 ページの「[リレーショナル表からの Price キューブのフェッチ](#)」でアナリティック・ワークスペースに追加してあるので、フラット・ファイルに追加のディメンション・メンバーが格納されていないかぎり、この 2 つのディメンションをメンテナンスする必要はありません。ただし、CUSTOMER ディメンションと CHANNEL ディメンションは、UNITS メジャーをロードする前に完全に移入する必要があります。

この例では、次の 3 つのフラット・ファイルからデータをロードします。

- CHANNEL.DAT には、CHANNEL ディメンションのすべてのメンバーおよびその属性が格納されています。このファイルは、GLOBAL スター・スキーマの CHANNEL_DIM ディメンション表（第3章を参照）と同等です。
- CUSTOMER.DAT には、CUSTOMER ディメンションのすべてのメンバーおよびその属性が格納されています。このファイルは、GLOBAL スター・スキーマの CUSTOMER_DIM ディメンション表（第3章を参照）と同等です。
- UNITS.DAT には、UNITS メジャーのベース・レベルのデータが格納されています。このファイルは、GLOBAL スター・スキーマの UNITS_HISTORY_FACT ファクト表（第3章を参照）と同等です。

フラット・ファイルからデータをロードする基本的なプロセスは、11-23 ページの「[リレーショナル表からの Price キューブのフェッチ](#)」で説明した、リレーショナル表からデータをロードするプロセスと同じです。唯一の違いは、OLAP DML プログラムの内容です。

CHANNELS.DAT からの Channels のロード

CHANNELS.DAT は、[例 11-14](#) に示すカンマ区切りファイルです。このファイルには、Global スター・スキーマの CHANNEL_DIM ディメンション表の各列に対応する次のフィールドがあります。

Channel ID
Channel Description
All Channels ID
All Channels Description

これらのフィールドを使用して、CHANNEL ディメンション、CHANNEL_LONG_DESCRIPTION 属性、CHANNEL_PARENTREL リレーションおよび CHANNEL_LEVELREL リレーションを移入できます。また、データ・ロード時に、CHANNEL_INHIER および CHANNEL_LEVELREL にリテラル・テキストを移入することもできます。

例 11-14 CHANNELS.DAT フラット・ファイル

```
2,Direct Sales,1,All Channels
3,Catalog,1,All Channels
4,Internet,1,All Channels
```

ディメンション値は、3 番目のフィールドにのみ記述される All Channels ディメンション・メンバー（1）を除き、通常の方法でロードできます。このディメンション・メンバーは、CHANNEL_PARENTREL の他のディメンション・メンバーの親として使用する前に、CHANNEL ディメンションに追加しておく必要があります。このため、3 番目のフィールドは、まず親を持たないディメンション・メンバーとして読み込み、再度親値として読み込みます。[例 11-15](#) は、データをロードするためのプログラムです。

サンプル・プログラムを定義し、次のコマンドでプログラムを実行します。


```
CALL read_channels
```

例 11-15 CHANNELS.DAT から Channels をロードする OLAP DML プログラム

```
DEFINE READ_CHANNELS PROGRAM
PROGRAM
VARIABLE funit INTEGER           "Define local variable for file handle
TRAP ON CLEANUP                 "Divert processing on error to CLEANUP label
funit = FILEOPEN('gx/channels.dat' READ)  "Open the file

FILEREAD funit CSV -
    FIELD 3 APPEND channel channel_inhier=yes channel_levelrel='ALL_CHANNELS' -
    FIELD 4 channel_long_description -
    FIELD 1 APPEND channel channel_inhier=YES channel_levelrel='CHANNEL' -
    FIELD 2 channel_long_description -
    FIELD 3 channel_parentrel

CLEANUP:
IF funit NE na
    THEN FILECLOSE funit
END
```

CHANNEL は 4 つのメンバーを持つだけの非常に小さなディメンションなので、サンプルを選択せずにロード結果を確認できます。例 11-16 にロード結果を示します。

例 11-16 CHANNEL ディメンションおよび属性の表示

```
REPORT W 8 DOWN channel W 12 <channel_long_description channel_parentrel
channel_levelrel channel_inhier>
```

```
ALL_LANGUAGES: AMERICAN_AMERICA
-----CHANNEL_HIERLIST-----
-----CHANNEL_ROLLUP-----
CHANNEL_LONG CHANNEL_PARE CHANNEL_LEVE CHANNEL_INHI
CHANNEL _DESCRIPTION NTREL LREL ER
-----
```

1	All Channels NA		ALL_CHANNELS	yes
2	Direct Sales 1		CHANNEL	yes
3	Catalog 1		CHANNEL	yes
4	Internet 1		CHANNEL	yes

CUSTOMERS.DAT からの Customers のロード

CUSTOMERS.DAT は、テキスト列が二重引用符で囲まれた構造化ファイルです。このファイルには、GLOBAL スター・スキーマの CUSTOMER_DIM ディメンション表の各列に対応する次のフィールドがあります。

Ship_To ID
Ship_To Description
Account ID
Account Description
Market Segment ID
Market Segment Description
Total Market ID
Total Market Description
Warehouse ID
Warehouse Description
Region ID
Region Description
All Customers ID
All Customers Description

例 11-17 は、サンプル・レコードの最初の 6 個のフィールドです。このファイルには CHANNELS.DAT と同じタイプの情報が格納されているので、同等のワークスペース・オブジェクトがすべて移入されます。大きな違いの 1 つは、データが 2 つの階層をサポートしていることです。

例 11-17 CUSTOMERS.DAT フラット・ファイル

49 "Bavarian Indust, GmbH Rome"	22 "Bavarian Industries"	5 "Manufacturing" ...
50 "Bavarian Indust, GmbH London"	22 "Bavarian Industries"	5 "Manufacturing" ...
55 "CiCi Douglas Chattanooga"	24 "CiCi Douglas"	5 "Manufacturing" ...
.		
.		
.		

CUSTOMERS.DAT のロード・プログラムは、CHANNELS.DAT のロード・プログラムと同様に、親ディメンション・メンバーをその子よりも先に読み込む必要があります。フィールド 13 には最大集計レベルである All Customers ID が格納されているので、これを最初にロードします。例 11-18 のプログラムは、SHIPMENTS_ROLLUP 階層の親メンバーを最初にロードし、次に MARKET_ROLLUP 階層の親メンバーをロードします。ベース・レベルである SHIP_TO は、これら両方の階層に属しています。

サンプル・プログラムを定義し、次のコマンドでプログラムを実行します。

CALL read_customers

例 11-18 CUSTOMERS.DAT を読み込む OLAP DML プログラム

```
DEFINE READ_CUSTOMERS PROGRAM
PROGRAM
VARIABLE funit INTEGER           "Define local variable for file handle
TRAP ON CLEANUP                 "Divert processing on error to CLEANUP label
funit = FILEOPEN('gx/customers.dat' READ) "Open the file
```

```

FILEREAD funit STRUCTURED -
  FIELD 13 APPEND customer -
    customer_inhier(customer_hierlist 'SHIPMENTS_ROLLUP')=yes -
    customer_levelrel='ALL_CUSTOMERS' -
  FIELD 14 customer_long_description(customer_hierlist 'SHIPMENTS_ROLLUP') -
  FIELD 11 APPEND customer -
    customer_inhier(customer_hierlist 'SHIPMENTS_ROLLUP')=yes -
    customer_levelrel='REGION' -
  FIELD 12 customer_long_description(customer_hierlist 'SHIPMENTS_ROLLUP') -
  FIELD 13 customer_parentrel(customer_hierlist 'SHIPMENTS_ROLLUP') -
  FIELD 9 APPEND customer -
    customer_inhier(customer_hierlist 'SHIPMENTS_ROLLUP')=yes -
    customer_levelrel='WAREHOUSE' -
  FIELD 10 customer_long_description(customer_hierlist 'SHIPMENTS_ROLLUP') -
  FIELD 11 customer_parentrel(customer_hierlist 'SHIPMENTS_ROLLUP') -
  FIELD 7 APPEND customer -
    customer_inhier(customer_hierlist 'MARKET_ROLLUP')=yes -
    customer_levelrel='TOTAL_MARKET' -
  FIELD 8 customer_long_description(customer_hierlist 'MARKET_ROLLUP') -
  FIELD 9 customer_parentrel(customer_hierlist 'MARKET_ROLLUP') -
  FIELD 5 APPEND customer -
    customer_inhier(customer_hierlist 'MARKET_ROLLUP')=yes -
    customer_levelrel='MARKET_SEGMENT' -
  FIELD 6 customer_long_description(customer_hierlist 'MARKET_ROLLUP') -
  FIELD 7 customer_parentrel(customer_hierlist 'MARKET_ROLLUP') -
  FIELD 3 APPEND customer -
    customer_inhier(customer_hierlist 'MARKET_ROLLUP')=yes -
    customer_levelrel='ACCOUNT' -
  FIELD 4 customer_long_description(customer_hierlist 'MARKET_ROLLUP') -
  FIELD 5 customer_parentrel(customer_hierlist 'MARKET_ROLLUP') -
  FIELD 1 APPEND customer -
    customer_inhier(customer_hierlist 'SHIPMENTS_ROLLUP')=yes -
    customer_inhier(customer_hierlist 'MARKET_ROLLUP')=yes -
    customer_levelrel='SHIP_TO' -
  FIELD 2 customer_long_description(customer_hierlist 'SHIPMENTS_ROLLUP') -
  FIELD 2 customer_long_description(customer_hierlist 'MARKET_ROLLUP') -
  FIELD 3 customer_parentrel(customer_hierlist 'MARKET_ROLLUP') -
  FIELD 9 customer_parentrel(customer_hierlist 'SHIPMENTS_ROLLUP')

CLEANUP:
IF funit NE na
  THEN FILECLOSE funit
END

```

CUSTOMER は非常に大きなディメンションなので、ロード結果をすべて表示することはできません。例 11-19 に、いくつかのベース・レベルのディメンションおよびその祖先を選択し

て、サポート・オブジェクトが正しく移入されているかどうかをチェックする方法を示します。

例 11-19 CUSTOMER ディメンションおよび属性の表示

```
LIMIT customer TO customer_levelrel 'SHIP_TO' "Select base-level members
LIMIT customer KEEP FIRST 2 "Keep the first 2 base-level members
LIMIT customer ADD ANCESTORS USING customer_parentrel "Add all their ancestors
SORT customer A customer_levelrel A CUSTOMER "Sort the selected members within levels

REPORT W 8 DOWN customer W 16 <customer_long_description customer_levelrel> -
      W 6 <customer_parentrel customer_inhier>

ALL_LANGUAGES: AMERICAN_AMERICA

-----CUSTOMER_HIERLIST-----
-----MARKET_ROLLUP-----SHIPMENTS_ROLLUP-----
CUSTOMER CUSTOMER CUSTOMER CUSTOMER CUSTOMER CUSTOMER CUSTOMER CUSTOMER
CUSTOMER_LONG_DE CUSTOMER_LEVELRE ER_PAR ER_INH CUSTOMER_LONG_DE CUSTOMER_LEVELRE ER_PAR ER_INH
CUSTOMER SCRIPTIO L ENTREL IER SCRIPTIO L ENTREL IER
-----
22 Bavarian ACCOUNT 5 yes NA ACCOUNT NA NA
Industries
1 All Customers ALL_CUSTOMERS NA NA All Customers ALL_CUSTOMERS NA yes
5 Manufacturing MARKET_SEGMENT 7 yes NA MARKET_SEGMENT NA NA
9 Europe REGION 1 NA Europe REGION 1 yes
49 Bavarian Indust, SHIP_TO 22 yes Bavarian Indust, SHIP_TO 16 yes
GmbH Rome GmbH Rome
50 Bavarian Indust, SHIP_TO 22 yes Bavarian Indust, SHIP_TO 20 yes
GmbH London GmbH London
7 Total Market TOTAL_MARKET 14 yes NA TOTAL_MARKET NA NA
14 Germany WAREHOUSE 9 NA Germany WAREHOUSE 9 yes
16 Italy WAREHOUSE 9 NA Italy WAREHOUSE 9 yes
20 United Kingdom WAREHOUSE 9 NA United Kingdom WAREHOUSE 9 yes
```

UNITS_CUBE.DAT ファイルの読み込み

UNITS_CUBE.DAT に格納されているのは、各ディメンション・キーの列を持つ Units メジャーのみです。例 11-20 にサンプル行を示します。

例 11-20 UNITS_CUBE.DAT フラット・ファイル

CHANNEL_ID	ITEM_ID	SHIP_TO_ID	MONTH_ID	UNITS
2	13	51	54	2
2	13	51	56	2
2	13	51	57	2
2	13	51	58	2

2	13	51	59	2
2	13	51	61	1
	.			
	.			
	.			

データは、UNITS 式ではなく UNITS_VARIABLE 変数に書き込まれます。
UNITS_VARIABLE の定義は次のとおりです。

```
DEFINE UNITS_VARIABLE VARIABLE DECIMAL <TIME UNITS_CUBE_COMPOSITE <CUSTOMER PRODUCT
CHANNEL>>
```

この変数は UNITS_CUBE_COMPOSITE によってディメンション化されていますが、[例 11-21](#)に示すように、入力データはベース・ディメンションに揃えて配置されます。4 つのベース・ディメンションは、すべて移入済です。

サンプル・プログラムを定義し、次のコマンドでプログラムを実行します。

```
CALL read_units
```

例 11-21 UNITS_CUBE.DAT を読み込む OLAP DML プログラム

```
DEFINE READ_UNITS PROGRAM
PROGRAM
VARIABLE funit INTEGER           "Define local variable for file handle
TRAP ON cleanup                 "Divert processing on error to cleanup label
funit = FILEOPEN('gx/units_cube.dat' READ)  "Open the file

FILEREAD funit STRUCTURED -
    FIELD 1 channel -
    FIELD 2 product -
    FIELD 3 customer -
    FIELD 4 time -
    FIELD 5 units_variable

cleanup:
IF funit NE na
    THEN FILECLOSE funit
END
```

通常、メジャーには大量のデータがスパースに格納されているので、データが正しくロードされたかどうかを確認するには、特定のセルを選択する必要があります。このためには、ソース・ファイルから 1、2 行選択し、[例 11-22](#)のように、ワークスペース・ディメンションをこれらの値に制限します。

例 11-22 UNITS_CUBE のデータ・ロードの検証

```
limit time to '50' to '60'
```

```
limit channel to '2'  
limit product to '13' '14'  
limit customer to '51'  
report down time across product: units_variable
```

```
CHANNEL: 2  
CUSTOMER: 51
```

---UNITS_VARIABLE---		
-----PRODUCT-----		
TIME	13	14
50	2.00	2.00
51	2.00	NA
52	2.00	2.00
53	1.00	2.00
54	2.00	2.00
55	NA	2.00
56	2.00	2.00
57	2.00	NA
58	2.00	1.00
59	2.00	2.00
60	NA	1.00

その他のスタンダード・フォームのメタデータ・オブジェクトの移入

ここで、GLOBALX アナリティック・ワークスペースを OLAP API に対して有効化すると、ディメンション・ビューには多くの空の列が含まれます。たとえば、CHANNEL ディメンションのビューには、次の空の列が含まれます。

```
CHANNEL_GID  
CHANNEL_PARENTGID  
ALL_CHANN_ALL_CHANNELS  
CHANNEL_CHANNEL  
AW_MEMBER_ORDER
```

____POP.FMLYREL プログラムおよび ____ORDR.HIERARCHIES プログラムは、これらの列によって表示されるワークスペース・オブジェクトを移入します。例 11-23 は、CHANNEL ディメンションに対して使用するコマンドです。これらのコマンドを、PRODUCT、CUSTOMER および TIME の各ディメンションに対しても実行します。

これらのオブジェクトを移入した後で、GLOBALX ワークスペースを再有効化する必要はありません。変更をデータベースにコミットすると、ただちにビューを介してデータを使用できるようになります。

例 11-23 CHANNEL メタデータ・オブジェクトを移入する OLAP DML コマンド

```
" Populate CHANNEL_GID and CHANNEL_FAMILYREL
```

```
CALL ____POP.FMLYREL('GLOBALX', 'GLOBALX!CHANNEL', 'GLOBALX!CHANNEL_HIERLIST',  
'GLOBALX!CHANNEL_LEVELLIST', 'GLOBALX!CHANNEL_LEVELREL', 'CHANNEL',  
'GLOBALX!CHANNEL_PARENTREL', 'GLOBALX!CHANNEL_INHIER')  
  
" Populate CHANNEL_ORDER  
call ____ordr.hierarchies('GLOBALX!CHANNEL', 'GLOBALX!CHANNEL_HIERLIST',  
'GLOBALX!CHANNEL_HIER_CREATEDBY', 'CHANNEL_PARENTREL', 'CHANNEL_ORDER',  
'CHANNEL_INHIER')
```

GLOBALX アナリティック・ワークスペースでのツールの使用

ここまでの作業が終了したら、集計プランの作成およびデプロイ・ウィザードおよびイネーブラを使用できます。

データをリフレッシュする際は、新しいデータソースにアクセスするようにデータ・ロード・プログラムを編集するか、またはロードを新しい期間に制限する必要があります。

第 IV 部

OLAP のデータベース管理

第 IV 部では、データベース管理者を対象に、Oracle OLAP に関連する管理作業について説明します。ここでは、次の項目について説明します。

- [第 12 章「Oracle OLAP の管理」](#)
- [第 13 章「OLAP API 用のマテリアライズド・ビュー」](#)

Oracle OLAP の管理

この章では、Oracle OLAP に関連する様々な管理作業について説明します。この章では、次の項目について説明します。

- 管理の概要
- アナリティック・ワークスペースの表領域の作成
- ユーザー名の設定
- Oracle OLAP の初期化パラメータ
- OLAP API の初期化パラメータ
- 外部ファイルへのアクセスの許可
- データ記憶域の理解
- パフォーマンスの監視

管理の概要

Oracle OLAP はデータベース内にあり、そのリソースは同じツールを使用して管理されるため、Oracle OLAP およびデータベースの管理は集中して行われます。しかし、データベース管理者またはアプリケーション開発者は、アナリティック・ワークスペースの作成およびメンテナンスの他にも、Oracle OLAP 固有の管理作業を行う必要があります。これらの作業を次に示します。

- 表領域。I/O ボトルネックを回避できるように、永続表領域および一時表領域を作成します（12-2 ページの「[アナリティック・ワークスペースの表領域の作成](#)」を参照）。
- データベース構成。パフォーマンスが最適化されるように、初期化パラメータを設定します（12-7 ページの「[Oracle OLAP の初期化パラメータ](#)」および 12-9 ページの「[OLAP API の初期化パラメータ](#)」を参照）。
- セキュリティ。OLAP アプリケーションのユーザーは、適切なアクセス権が付与されたデータベース識別情報を持っている必要があります。ユーザーがファイルにアクセスで

きるようにするには、データベース・ディレクトリ・オブジェクトを定義し、これに対するアクセス権をユーザーに付与する必要があります（12-5 ページの「[ユーザー名の設定](#)」を参照）。

- パフォーマンス。データベース監視ツールを使用すると、過去の使用状況に基づいて、データベース構成に加えることが推奨される変更を確認できます（12-13 ページの「[パフォーマンスの監視](#)」を参照）。

参照： Oracle Database の管理の詳細は、『Oracle Database 管理者ガイド』を参照してください。

アナリティック・ワークスペースの表領域の作成

アナリティック・ワークスペースを作成する前に、アナリティック・ワークスペース専用の UNDO 表領域、永続表領域および一時表領域を作成する必要があります。アナリティック・ワークスペースは、特に指定しないかぎり、ユーザーのデフォルトの表領域に作成されます。デフォルトの表領域は、すべてのユーザーについて、初期状態では SYS に設定されています。アナリティック・ワークスペースを SYS 表領域に作成すると、全体的なパフォーマンスが低下する可能性があります。また、アナリティック・ワークスペースの表領域は、リレーショナル表（特に、ソースのスター・スキーマまたはスノーフレーク・スキーマ）と共有にするべきではありません。

Oracle OLAP では一時表領域を頻繁に使用するため、I/O ボトルネックを回避できるように正確に設定することが特に重要です。

Oracle OLAP 用に設定した表領域は、アナリティック・ワークスペースによって使用される他に、OLAP カタログ・メタデータやワークスペース・データのビューの作成およびメンテナンスなどの作業で SQL によっても使用されます。

できるだけ多くのコントローラおよびドライブで、データファイルおよび一時ファイルをストライプ化してください。

UNDO 表領域の作成

UNDO 表領域を作成するには、次の SQL コマンドを発行します。

```
CREATE UNDO TABLESPACE tablespace DATAFILE 'pathname'  
    SIZE size REUSE AUTOEXTEND ON NEXT size  
    MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL;
```

各項目の意味は次のとおりです。

tablespace は表領域の名前です。
pathname は完全修飾されたファイル名です。
size は適切なバイト数です。

次に例を示します。

```
CREATE UNDO TABLESPACE olapundo DATAFILE '$ORACLE_HOME/oradata/undo.dbf'
  SIZE 64M REUSE AUTOEXTEND ON NEXT 8M
  MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL;
```

UNDO 表領域を作成したら、これらの設定をシステム・パラメータ・ファイルに反映し、データベースを再起動します（12-7 ページの「[Oracle OLAP の初期化パラメータ](#)」を参照）。

```
UNDO_TABLESPACE=tablespace
UNDO_MANAGEMENT=AUTO
```

アナリティック・ワークスペースの永続表領域の作成

ユーザーが作成したアナリティック・ワークスペースは、そのユーザーのデフォルトの表領域に作成されます。デフォルトの表領域は、初期状態では SYS 表領域に設定されています。次の SQL 文は、アナリティック・ワークスペースの格納に適した表領域を作成します。

```
CREATE TABLESPACE tablespace DATAFILE 'pathname'
  SIZE size REUSE AUTOEXTEND ON NEXT size MAXSIZE UNLIMITED
  EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

```
ALTER USER username DEFAULT TABLESPACE tablespace
```

各項目の意味は次のとおりです。

tablespace は表領域の名前です。

pathname は完全修飾されたファイル名です。

size は適切なバイト数です。

username はデータベース・ユーザーの名前です。

次に例を示します。

```
CREATE TABLESPACE glo DATAFILE '$ORACLE_HOME/oradata/glo.dbf'
  SIZE 64M REUSE AUTOEXTEND ON NEXT 8M MAXSIZE UNLIMITED
  EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

コンピュータに複数のディスクがある場合は、表領域をこれらのディスクでストライプ化できます。次の SQL 文は、GLO 表領域を 3 つの物理ディスクに分散します。

```
CREATE TABLESPACE glo DATAFILE
  'disk1/oradata/glo1.dbf' SIZE 64M REUSE AUTOEXTEND ON NEXT 8M MAXSIZE 1024M
  EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

ALTER TABLESPACE glo ADD DATAFILE
  'disk2/oradata/glo2.dbf' SIZE 64M REUSE AUTOEXTEND ON NEXT 8M MAXSIZE 1024M,
  'disk3/oradata/glo3.dbf' SIZE 64M REUSE AUTOEXTEND ON NEXT 8M MAXSIZE UNLIMITED;
```

アナリティック・ワークスペースの一時表領域の作成

Oracle OLAP は、アナリティック・ワークスペース内のデータに対するすべての変更（データ・ロード、what-if 分析、予測、集計、またはその他の分析の結果によるあらゆる変更）を格納するのに一時表領域を使用します。OLAP DML の UPDATE コマンドにより、これらの変更が永続表領域に移動され、一時表領域がクリアされます。

Oracle OLAP はまた、一時表領域を使用して異なる世代のアナリティック・ワークスペースをメンテナンスします。これによって、内容の更新中に 1 人以上のユーザーがそのビューを読み込んでいる場合でも、アナリティック・ワークスペースの一貫性のあるビューを表示できます。このため表領域内には多数の拡張領域が作成されるので、EXTENT MANAGEMENT には小さいサイズを指定するようにします。

次のコマンドは、Oracle OLAP での使用に適した一時表領域を作成します。

```
CREATE TEMPORARY TABLESPACE tablespace TEMPFILE 'pathname'  
    SIZE size REUSE AUTOEXTEND ON NEXT size MAXSIZE UNLIMITED  
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE size;
```

各項目の意味は次のとおりです。

pathname は完全修飾されたファイル名です。

size は適切なバイト数です。

tablespace は表領域の名前です。

username はデータベース・ユーザーの名前です。

次に例を示します。

```
CREATE TEMPORARY TABLESPACE glotmp TEMPFILE '$ORACLE_HOME/oradata/glotmp.tmp'  
    SIZE 50M REUSE AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED  
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
```

永続表領域と同じように、一時表領域も複数のディスクでストライプ化できます。次の SQL 文は、GLOTMP 一時表領域を 3 つの物理ディスクでストライプ化します。

```
CREATE TEMPORARY TABLESPACE glotmp TEMPFILE  
    'disk1/oradata/glotmp1.tmp' SIZE 50M REUSE AUTOEXTEND ON NEXT 5M MAXSIZE 1024M  
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;  
  
ALTER TABLESPACE glotmp ADD TEMPFILE  
    'disk2/oradata/glotmp2.tmp' SIZE 50M REUSE AUTOEXTEND ON NEXT 5M MAXSIZE 1024M,  
    'disk3/oradata/glotmp3.tmp' SIZE 50M REUSE AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED;
```

アナリティック・ワークスペースのサイズの問合せ

特定のアナリティック・ワークスペースの表領域拡張のサイズを確認するには、次の SQL 文を使用します。

```
COLUMN DBMS_LOB.GETLENGTH(AWLOB) HEADING "Bytes";
```

```
SELECT EXTNUM, DBMS_LOB.GETLENGTH(AWLOB) FROM AW$auname;
```

各項目の意味は次のとおりです。

awname はアナリティック・ワークスペースの名前です。

ユーザー名の設定

データベースに接続するには、データベース・セキュリティによって認証可能なユーザー名とパスワードを指定する必要があります。すべてのユーザーは、CONNECT ロールを持っている必要があります。ユーザー名に関連付けられたその他の権限によって、データへのユーザー・アクセスが制御されます。データベース管理者は、Oracle OLAP のすべてのユーザーに、適切な資格証明を持つユーザー名を設定する必要があります。

定義したユーザー名に、Oracle Enterprise Manager のセキュリティ・フォルダからこれらの権限を付与できます。また、SQL コマンドを使用することもできます。

データベースのインストール時に、Oracle OLAP をサポートする次の 2 つのロールが明示的に定義されます。

- OLAP_USER ロールは、OLAP カタログ・メタデータおよびスタンダード・フォームのアナリティック・ワークスペースを自身のスキーマに作成して管理するための権限をユーザーに付与します。これらの作業を行う OLAP ユーザーは、OLAP_USER ロールまたは同等の権限を持っている必要があります。
- OLAP_DBA ロールは、OLAP カタログ・メタデータおよびスタンダード・フォームのアナリティック・ワークスペースを任意のスキーマに作成して管理するための権限を DBA またはシステム管理者に付与します。OLAP_DBA ロールは、DBA ロールとともに付与されます。この権限を管理者以外のユーザーに付与する場合は注意が必要です。

参照： 権限の付与の詳細は、『Oracle9i SQL リファレンス』を参照してください。

DBA およびアプリケーション開発者の SQL アクセス

OLAP カタログ・メタデータを作成するユーザーは、OLAP_USER ロールを付与されている必要があります。アナリティック・ワークスペースを作成するユーザーには、ソース・スキーマ表に対する SELECT 権限と、ワークスペースが作成される表領域に対する無制限の割当て制限も必要です。例 12-1 は、GLOBAL_AW ユーザーを作成する SQL 文です。

例 12-1 GLOBAL_AW ユーザーを作成する SQL 文

```
CREATE USER 'GLOBAL_AW' IDENTIFIED BY 'global_aw'
  DEFAULT TABLESPACE glo
  TEMPORARY TABLESPACE glotmp
  QUOTA UNLIMITED ON glo
  QUOTA UNLIMITED ON glotmp
```

```
ACCOUNT UNLOCK;

GRANT SELECT ON global.channel_dim TO global_aw;
GRANT SELECT ON global.customer_dim TO global_aw;
GRANT SELECT ON global.product_dim TO global_aw;
GRANT SELECT ON global.time_dim TO global_aw;
GRANT SELECT ON global.price_and_cost_history_fact TO global_aw;
GRANT SELECT ON global.price_and_cost_update_fact TO global_aw;
GRANT SELECT ON global.units_history_fact TO global_aw;
GRANT SELECT ON global.units_update_fact TO global_aw;
```

分析者の SQL アクセス

既存のアナリティック・ワークスペースにアクセスするユーザーは、ワークスペースが格納されている表に対して次のアクセス権を持っている必要があります。

- アナリティック・ワークスペースから読み込みを行う場合は SELECT 権限
- アナリティック・ワークスペースに書き込みを行う場合は SELECT、INSERT および UPDATE の各権限

表の名前は、アナリティック・ワークスペースの名前に接頭辞「AW\$」を付け加えたものになります。たとえば、XADEMO というアナリティック・ワークスペースは、AW\$XADEMO というリレーショナル表に格納されます。

ワークスペース・データのビューにアクセスするユーザーは、それらのビューに対する EXECUTE 権限を明示的に付与されている必要があります。

例 12-2 の SQL 文は、GLOBAL アナリティック・ワークスペースに対する読み込み専用権限をすべてのユーザーに付与し、読み込み / 書き込み権限をユーザー SCOTT に付与します。

例 12-2 GLOBAL アナリティック・ワークスペースに対するアクセス権の付与

```
GRANT SELECT ON global_aw.aw$ global TO PUBLIC;
GRANT INSERT ON global_aw.aw$ global TO scott;
GRANT UPDATE ON global_aw.aw$ global TO scott;
```

OLAP API を使用したデータベース・オブジェクトへのアクセス

OLAP API を使用してデータベースに接続するには、データベースに対する次のアクセス権が必要です。

- CONNECT ロール。
- QUERY REWRITE システム権限。
- 分析対象のデータを格納するデータベース・オブジェクト（アナリティック・ワークスペースまたはリレーショナル表）に対する SELECT 権限。アナリティック・ワークス

ペースに対するアクセス権の付与については、前述の「[分析者の SQL アクセス](#)」を参照してください。

Oracle OLAP の初期化パラメータ

表 12-1 に、Oracle OLAP のパフォーマンスに影響するパラメータを示します。サーバー・パラメータ・ファイルまたは `init.ora` ファイルをこれらの値に変更した後、データベース・インスタンスを再起動します。これらの設定の効果を監視し、必要に応じて設定を調整できます。

表 12-1 データベース・パラメータ・ファイルの初期設定

パラメータ	設定
DB_CACHE_SIZE	物理メモリー /2。
OLAP_PAGE_POOL_SIZE	問合せ用には 2 ～ 8MB。データ・ロードを行う場合は一時的に増やします。
PARALLEL_MAX_SERVERS	プロセスの数 -1。 このパラメータは、パラレル更新、およびリレーショナル表から読み込みを行う際の SQL の SELECT 操作で使用されるプロセスの数を制限します。また、パラレル処理の数は、アナリティック・ワークスペースの更新される拡張ファイルの数に依存します。
PGA_AGGREGATE_TARGET	200 ～ 400MB。
SESSIONS	2.5 × 同時 OLAP ユーザーの最大数。
UTL_FILE_DIR	Oracle Database がファイルへの書き込みを行うことができるディレクトリ・パス。
UNDO_MANAGEMENT	AUTO
UNDO_TABLESPACE	UNDO 表領域の名前。UNDO 表領域は先に定義しておく必要があります (12-2 ページの「 UNDO 表領域の作成 」を参照)。

参照： これらのパラメータについては、『Oracle Database パフォーマンス・チューニング・ガイド』を参照してください。

手順 : OLAP 用システム・パラメータの設定

次の手順でシステム・パラメータを設定します。

1. テキスト・エディタで `initsid.ora` 初期化ファイルを開きます。

初期化ファイルは、`$ORACLE_HOME/admin/sid/pfile`（この場合の `sid` は、`$ORACLE_HOME/network/admin/tnsnames.ora` で定義されたシステム識別子）に格納されています。

2. ファイルの設定を追加または変更します。

たとえば、Oracle が `olapscripts` ディレクトリにファイルを書き込むことができるようにするには、次のように記述します。

```
UTL_FILE_DIR=/users/oracle/olapscripts
```

3. 次のコマンドを使用して、データベースを停止および再起動します。STARTUP コマンドの初期化ファイルが正しいものであることを確認してください。

```
SQLPLUS '/ AS SYSDBA'
SHUTDOWN IMMEDIATE
STARTUP pfile=$ORACLE_HOME/admin/re110i/pfile/initre110i.ora
```

OLAP_PAGE_POOL_SIZE の設定について

OLAP_PAGE_POOL_SIZE は Oracle OLAP 固有の初期化パラメータです。このパラメータは、各 OLAP セッションに割り当てるページ・プールの最大サイズをバイト単位で指定します。OLAP_PAGE_POOL_SIZE の最小値は 4MB です。デフォルト値は 32MB です。

OLAP_PAGE_POOL_SIZE の設定に関する基本的なガイドラインを次に示します。

- データベース初期化ファイルで、同時 OLAP ユーザーの最大数に基づいて OLAP_PAGE_POOL_SIZE の値を設定します。このパラメータは、2 ～ 8MB 程度に設定します。通常は 4MB です。数値を大きくするとパフォーマンスが向上しますが、指定した容量が各ユーザーに割り当てられることに注意してください。
- データ・ロードを行う場合は、SQL の ALTER SESSION 文を使用して、ロードの間だけ OLAP ページ・プールのサイズをできるかぎり大きくします。データ・ロードを行っている間は、ページ・プールは他のユーザーと共有されません。100 ～ 400MB 程度で、DB_CACHE_SIZE よりも小さくなるように設定します。

OLAP ページ・プールは OLAP セッションの開始時に割り当てられ、ユーザーがセッションを終了したときに解放されます。OLAP セッションは OLAP_TABLE ファンクション、PL/SQL パッケージ DBMS_AWM または OLAP Worksheet のコマンドラインで開始できます。

OLAP ページ・プールはユーザー・グローバル領域（UGA）から割り当てられます。データベースが専用モードで実行されている場合、UGA はプロセス・グローバル領域（PGA）の一部です。データベースが共有サーバー・プロセスとして実行されている場合、UGA は共有グローバル領域（SGA）の一部です。

OLAP ページ・プールが満杯になると、スワップ領域として DB キャッシュが使用されます。これはメモリー内スワップなので、処理は比較的高速です。DB キャッシュが満杯になると、キャッシュがディスクにスワップされます。この処理は比較的低速です。DB キャッシュを頻繁にディスクにスワップしなければならなくなると、パフォーマンスが大幅に低下します。

PGA_AGGREGATE_TARGET の設定について

PGA_AGGREGATE_TARGET は、SQL 文、特に GROUP BY 句および ORDER BY 句を付けた SELECT 文を実行する際に使用されます。OLAP エンジンはこのパラメータを使用しません。ただし、リレーショナル表からデータを選択する際に、PGA_AGGREGATE_TARGET が OLAP API のパフォーマンスに影響を与える場合があります。Oracle Database でこのタイプのアプリケーションをサポートしている場合は、まず PGA_AGGREGATE_TARGET を 200 ～ 400MB に設定してから、データベース・パフォーマンスの監視ツールを使用して調整が必要かどうかを確認します。

OLAP API の初期化パラメータ

OLAP API は、データベースの構成パラメータが OLAP 用に最適化されている場合に、最適なパフォーマンスを提供します。Oracle Database のインストール時に、OLAP 構成表が作成され、OLAP API のパフォーマンスが最適になるようにテスト済の ALTER SESSION コマンドが移入されます。OLAP API がセッションを開くたびに、これらの ALTER SESSION コマンドが実行されます。

データベース・インスタンスが OLAP API を使用する Java アプリケーションのみをサポートするために使用されている場合は、サーバー・パラメータ・ファイルまたは init.ora ファイルを変更して、これらの設定を含めることができます。または、今後のデータベース・インスタンスの使用方法に応じて、サーバー・パラメータ・ファイルに一部の設定を含め、その他の設定を構成表に残すこともできます。次のような選択肢があります。

- すべてのパラメータを構成表に残します。この場合、それらのパラメータは OLAP API セッションの初期化の一部として設定されます。この方法では、これらの構成手順は OLAP API 用に完全に切り離されます。(デフォルト)
- 構成パラメータの一部をサーバー・パラメータ・ファイルまたは init.ora ファイルに追加し、それらの行を構成表から削除します。この方法は、同じ設定が必要な他のアプリケーションでデータベースが使用されている場合に有効です。
- すべての構成パラメータをサーバー・パラメータ・ファイルまたは init.ora ファイルに追加し、すべての行を構成表から削除します。この方法は、データベース・インスタンスが OLAP API のみで使用されている場合に最も便利です。

これらのパラメータの設定場所にかかわらず、最新の推奨事項について Oracle Technology Network を参照してください。

参照： ALTER SESSION コマンドで設定できる初期化パラメータについては、『Oracle9i SQL リファレンス』を参照してください。

外部ファイルへのアクセスの許可

OLAP DML には、外部ファイルに対して読み込みおよび書き込みを実行する 3 種類のコマンドが含まれています。

- フラット・ファイルからアナリティック・ワークスペース・オブジェクトにデータをコピーするファイル読み込みコマンド。
- アナリティック・ワークスペース・オブジェクトとその内容をファイルにコピーし、別のデータベース・インスタンスに転送するインポートおよびエクスポート・コマンド。
- DML コマンドをファイルから読み込んで実行し、コマンド出力をファイルにリダイレクトするファイル入力および出力コマンド。

これらのコマンドは、BFILE セキュリティを使用してファイルへのアクセスを制御します。このデータベース・セキュリティ・メカニズムによって、物理ディスク・ディレクトリを表す論理データベース・ディレクトリが作成されます。権限がデータベース・ディレクトリに割り当てられ、この権限によって、関連付けられた物理ディレクトリのファイルへのアクセスが制御されます。

データベース・ディレクトリを作成し、権限を付与するには、PL/SQL 文を使用します。これらの SQL 文の関連する構文について次に説明します。

参照： 完全な構文および使用上の注意については、『Oracle9i SQL リファレンス』の CREATE DIRECTORY および GRANT の説明を参照してください。

データベース・ディレクトリの作成

データベース・ディレクトリを作成するには、CREATE ANY DIRECTORY システム権限が必要です。

次の PL/SQL 構文を使用して、CREATE DIRECTORY 文で新しいディレクトリを作成するか、REPLACE DIRECTORY 文で既存のディレクトリを再定義します。

```
{CREATE | REPLACE | CREATE OR REPLACE} DIRECTORY directory AS 'pathname';
```

各項目の意味は次のとおりです。

directory は論理データベース・ディレクトリの名前です。

pathname は物理ディレクトリのパスです。

データベース・ディレクトリに対するアクセス権の付与

ディレクトリを作成した後、次の PL/SQL 構文を使用して、そのディレクトリに含まれるファイルに対するアクセス権をユーザーおよびグループに付与します。

```
GRANT permission ON DIRECTORY directory TO {user | role | PUBLIC};
```

各項目の意味は次のとおりです。

permission は次のいずれかです。

読み専用アクセスの場合は READ

書き専用アクセスの場合は WRITE

読み書き両用アクセスの場合は ALL

directory はデータベース・ディレクトリの名前です。

user は、*directory* に対する直接アクセス権を取得するデータベース・ユーザーです。

role は、*directory* に対する直接アクセス権を取得するデータベース・ロールです。

PUBLIC を指定すると、すべてのデータベース・ユーザーに *directory* に対する直接アクセス権が付与されます。

例：データベース・ディレクトリの作成と使用

次の SQL コマンドは、ディレクトリ `/users/oracle/OraHome1/olap` へのアクセスを制御するディレクトリ OLAPFILES を作成し、すべてのユーザーに読込みアクセス権を付与します。

```
CREATE DIRECTORY olapfiles as '/users/oracle/OraHome1/olap';  
GRANT READ ON DIRECTORY olapfiles TO PUBLIC;
```

ユーザーは、次のような DML コマンドを使用して、`/users/oracle/OraHome1/olap` に格納されているファイルにアクセスします。

```
IMPORT ALL FROM EIF FILE 'olapfiles/salesq2.eif' DATA DFNS
```

データ記憶域の理解

Oracle OLAP の多次元データはアナリティック・ワークスペースに格納され、アナリティック・ワークスペースはリレーショナル表に格納されます。アナリティック・ワークスペースには、ディメンション、変数（メジャー）、OLAP DML プログラムなど、様々なオブジェクトを格納できます。通常、これらのオブジェクトは特定のアプリケーションまたはデータ・セットをサポートします。

アナリティック・ワークスペースを作成または変更したり、アナリティック・ワークスペースにアクセスすると、必ずリレーショナル・データベースの表に情報が格納されます。

重要： これらの表は Oracle OLAP の操作に必須です。これらの表を削除または変更した場合の影響を十分に理解していない場合は、削除したり直接変更しないでください。

ユーザーが所有する表

アナリティック・ワークスペースは、Oracle Database の表にバイナリ・ラージ・オブジェクト（BLOB）として格納されます。

たとえば、GLOBAL_AW ユーザーが GLOBAL と GLOBAL_PROGRAMS という 2 つのアナリティック・ワークスペースを作成すると、次の表が GLOBAL スキーマに作成されます。

```
AW$GLOBAL
AW$GLOBAL_PROGRAMS
```

これらの表には、これらのアナリティック・ワークスペースのすべてのオブジェクト定義およびデータが格納されます。

システム表

SYS ユーザーはアナリティック・ワークスペースに関連付けられた複数の表を所有します。

```
AW$EXPRESS
AW$AWCREATE
AW$AWMD
AW$
PS$
```

- AW\$EXPRESS には EXPRESS アナリティック・ワークスペースが格納されます。このアナリティック・ワークスペースには、OLAP DML をサポートするオブジェクトおよびプログラムが含まれます。EXPRESS アナリティック・ワークスペースは、セッションが開いているときに使用されます。
- AW\$AWCREATE には AWCREATE アナリティック・ワークスペースが格納されます。このアナリティック・ワークスペースには、スタンダード・フォームのアナリティック・ワークスペースを作成および管理するプログラムが含まれます。
- AW\$AWMD には AWMD アナリティック・ワークスペースが格納されます。このアナリティック・ワークスペースには、スタンダード・フォームのカタログを作成するプログラムが含まれます。
- AW\$ はデータベースのすべてのアナリティック・ワークスペースのレコードを保持し、レコードの名前、所有者およびその他の情報を記録します。
- PS\$ はすべてのページ領域の履歴を保持します。ページは順序付けられた一連のバイトで、ファイルに相当します。Oracle OLAP は、アナリティック・ワークスペース・ページのキャッシュを管理します。ページは表の記憶域から読み込まれ、問合せへの応答でキャッシュに書き込まれます。複数のセッションで同じページにアクセスすることもできます。

1 つのアナリティック・ワークスペースに対して、同時に 1 回の書込みと多数の読み込みを実行することができます。PS\$ に格納された情報を使用すると、セッション中にアナリティック・ワークスペースが変更されている場合でも、Oracle OLAP は不要になったページを廃棄し、すべてのユーザーのデータのビューを一貫性のある状態にメンテナンスできます。アナリティック・ワークスペースへの変更が保存されると、未使用のページは削除され、対応する行は PS\$ から削除されます。

CWM1 および CWM2 読み込み API は、OLAPSYS ユーザーが所有する表です。パブリック・シノニムを使用することで、ユーザーがこれらの表にアクセスできるようになります。

パフォーマンスの監視

各 Oracle Database インスタンスは、現行のデータベース・アクティビティを記録する一連の仮想的な表を保持します。これらの表を動的パフォーマンス表と呼びます。動的パフォーマンス表は、内部ディスク構造およびメモリー構造のデータを収集します。その中には、Oracle OLAP のデータを収集する表もあります。これらの表を監視すると、使用方法の傾向を発見し、システムのボトルネックを診断できます。OLAP 動的パフォーマンス・ビューについては、『Oracle OLAP リファレンス』を参照してください。

OLAP API 用のマテリアライズド・ビュー

この章では、OLAP API の要件に固有のマテリアライズド・ビューを作成する方法について説明します。アナリティック・ワークスペースを使用している場合は、この章は読まなくてもかまいません。アナリティック・ワークスペースは、マテリアライズド・ビューを必要としない形で集計データを生成および格納するためです。ただし、完全にリレーショナルなアプリケーションを開発している場合は、この章で説明する方法でマテリアライズド・ビューを作成する必要があります。さもないと、マテリアライズド・ビューを作成するために使用される SQL が OLAP API によって生成される SQL と一致せず、問合せに対する回答を作成する際にクエリー・リライトがマテリアライズド・ビューを使用しなくなります。

参照： マテリアライズド・ビューの管理については、『Oracle データ・ウェアハウス・ガイド』を参照してください。

この章では、次の項目について説明します。

- [Oracle OLAP でのサマリー管理](#)
- [概要と要件](#)
- [例：ディメンションのマテリアライズド・ビュー](#)
- [例：ファクトのマテリアライズド・ビュー](#)
- [DBMS_ODM パッケージの使用](#)

Oracle OLAP でのサマリー管理

オンライン分析処理（OLAP）の基本的な機能は、様々なレベルの集計データを分析および表示できることです。Oracle OLAP では、集計データをアナリティック・ワークスペースとマテリアライズド・ビューのどちらに格納するかを選択できます。

リレーショナル・ウェアハウスのサマリー管理は、Oracle のクエリー・リライト機能によって管理されます。クエリー・リライトを使用すると、集計の実行時に再計算するのではなく、問合せで集計データをマテリアライズド・ビューからフェッチできます。

OLAP API がリレーショナル表に格納されているウェアハウスを問い合わせる場合、可能な場合はクエリー・リライトが使用されます。OLAP API によるアクセス用にリレーショナル・ウェアハウスを準備するには、この章で後述するガイドラインに従って、マテリアライズド・ビューを作成する必要があります。

概要と要件

OLAP API は、スター・スキーマにマップされる OLAP カタログ・キューブごとに、専用のマテリアライズド・ビューのセットを必要とします。キューブは単一のファクト表にマップされ、ファクト表は最低レベルのデータのみを含む必要があります。

各キューブに対しては、キューブの各ディメンションの階層ごとに、個別のディメンションのマテリアライズド・ビューが必要です。キューブのファクト表に対しては、GROUP BY GROUPING SETS 構文で作成された単一のマテリアライズド・ビューが必要です。

マテリアライズド・ビューを作成するには、Oracle Data Management パッケージの DBMS_ODM を使用します。

重要： OLAP API 用のマテリアライズド・ビューを作成するのに、DBMS_OLAP パッケージは使用しないでください。クエリー・リライトは、OLAP API によって生成された SQL を、このパッケージで生成されたマテリアライズド・ビューにマップすることはしません。

DBMS_OLAP パッケージについては、『Oracle データ・ウェアハウス・ガイド』を参照してください。

キューブに必要なマテリアライズド・ビュー

OLAP API は、キューブに関連付けられた階層ごとにディメンションのマテリアライズド・ビューを必要とします。たとえば、Sales History (SH) スキーマ内の SALES_CUBE キューブには、7 個のディメンションのマテリアライズド・ビューが必要です (表 13-1 を参照)。

キューブのファクト表については、OLAP API は単一のグルーピング・セット形式のマテリアライズド・ビューを必要とします。

表 13-1 SH.SALES_CUBE のディメンションのマテリアライズド・ビューの数

SALES_CUBE ディメンション	階層	マテリアライズド・ビューの数
SH.CHANNELS_DIM	CHANNEL_ROLLUP	1
SH.CUSTOMERS_DIM	CUST_ROLLUP	2
	GEOG_ROLLUP	
SH.PRODUCTS_DIM	PROD_ROLLUP	1
SH.PROMOTIONS_DIM	PROMO_ROLLUP	1
SH.TIMES_DIM	CAL_ROLLUP	2
	FIS_ROLLUP	

マテリアライズド・ビューおよび OLAP メタデータ

マテリアライズド・ビューを作成する前に、スター・スキーマの OLAP メタデータを作成する必要があります。Oracle Enterprise Manager を使用するか、または CWM2 パッケージを使用するスクリプトを記述します。OLAP カタログについては、[第 5 章](#)を参照してください。

例：ディメンションのマテリアライズド・ビュー

ディメンションのマテリアライズド・ビューを作成する SQL スクリプトには、CREATE MATERIALIZED VIEW 文と統計およびビットマップ索引を生成する文が含まれます。

ディメンション階層のマテリアライズド・ビューの作成

ディメンション階層用の CREATE MATERIALIZED VIEW 文の基本構文は次のとおりです。

```
CREATE MATERIALIZED VIEW mv_name
PARTITION BY RANGE (gid)
(partition values less than(1) ,
.
.
partition values less than(MAXVALUE))
TABLESPACE tblspace_name
BUILD IMMEDIATE
USING NO INDEX
REFRESH FORCE
ENABLE QUERY REWRITE
AS
SELECT
COUNT(*) COUNT_STAR,
GROUPING_ID(level_columns) gid,
```

```
MAX(attribute_column_1)
.
.
MAX(attribute_column_n)
level_cols
FROM
dimension_tables
GROUP BY
hierarchy1_level1, ROLLUP(hierarchy1_level2,... hierarchy1_leveln),
hierarchy2_level1, ROLLUP(hierarchy2_level2,... hierarchy2_leveln),
.
.
hierarchy_n_level1, ROLLUP(hierarchy_n_level2,... hierarchy_n_leveln);
```

GROUP BY 句には、最大集計レベル（level1）から最小集計レベル（leveln）の順にレベル列が指定されます。最小集計レベル（リーフ・ノード）は、キー列でもあります。level1 は、ROLLUP リストから除外されることに注意してください。

ディメンション階層のビットマップ索引

スクリプトには、レベル列および GID 列のビットマップ索引を生成する次のような文が含まれます。スクリプトは、親 GID および親 ET キーのビットマップ索引も計算します。

```
CREATE BITMAP INDEX index_name ON mv_name(level_column)
PCTFREE 0
COMPUTE STATISTICS
LOCAL
NOLOGGING;
```

ディメンション階層の統計

スクリプトには、統計を生成する次のような文が含まれます。

```
execute dbms_stats.gather_table_stats(mv_owner, mv_name,
degree=>dbms_stats.default_degree,method_opt=>
'for all columns size skewonly') ;
ALTER TABLE mv_name MINIMIZE RECORDS_PER_BLOCK ;
```

例：ファクトのマテリアライズド・ビュー

DBMS_ODM パッケージによって生成される、ファクトのマテリアライズド・ビューを作成するための SQL スクリプトには、CREATE MATERIALIZED VIEW 文と統計およびビットマップ索引を生成する文が含まれます。

ファクトのマテリアライズド・ビューの作成

グルーピング・セットを使用する、ファクト表用の CREATE MATERIALIZED VIEW 文の基本構文は次のとおりです。

```
CREATE MATERIALIZED VIEW mv_name
PARTITION BY RANGE (gid)
    (partition values less than(1) ,
     .
     .
     partition values less than(MAXVALUE))
PCTFREE x PCTUSED y
BUILD IMMEDIATE
USING NO INDEX
REFRESH FORCE
ENABLE QUERY REWRITE
AS
SELECT
    GROUPING_ID(level_columns) gid,
    agg_method(measure_1),
    .
    .
    agg_method(measure_n),
    COUNT(*) COUNT_OF_STAR,
    level_columns
FROM
    dimension_tables, fact_table
WHERE
    (dimension_primary_key_1 = fact_foreign_key_1) AND
    .
    .
    (dimension_primary_key_n = fact_foreign_key_n)
GROUP BY GROUPING SETS (
    (level columns in grouping set_1),
    .
    .
    (level columns in grouping set_n);
```

各グルーピング・セットには、集計用に指定したレベルの組合せが含まれます。たとえば、グルーピング・セットで、キューブのデータを各地域の全製品について月別で集計するように指定できます。DBMS_ODM パッケージのプロシージャは、SYS.OLAPTABLELEVELS と SYS.OLAPTABLELEVELTUPLES という 2 つの表を使用して、各グルーピング・セットにレベルの組合せを作成します。これらの表の生成および編集については、13-6 ページの「[手順：グルーピング・セット形式のマテリアライズド・ビューの作成](#)」を参照してください。

SELECT 句には、ディメンション表のレベルとファクト表のメジャーが指定されます。選択されたメジャーは、集計用に指定されたこれらのレベルの各組合せに基づいて集計されます。通常、集計メソッドは加算 (SUM) ですが、平均や加重平均にすることもできます。各

メジャーに関連付けられる集計メソッドは、メジャーの OLAP カタログ・メタデータで指定されています。

ファクトのマテリアライズド・ビューのビットマップ索引

スクリプトには、マテリアライズド・ビューに含める対象として選択した各レベルのビットマップ索引を生成する次のような文が含まれます。スクリプトは、ディメンション内のすべての上位集計レベルのビットマップ索引も作成します。たとえば、時間カレンダー階層の四半期レベルに集計するように選択した場合は、年と四半期のビットマップ索引が作成されません。

```
CREATE BITMAP INDEX index_name ON mv_name(level_col)
LOCAL
COMPUTE STATISTICS
PARALLEL PCTFREE 0
NOLOGGING;
```

ファクトのマテリアライズド・ビューの統計

スクリプトには、統計を生成する次のような文が含まれます。

```
execute dbms_stats.gather_table_stats(mv_owner, mv_name,
    degree=>dbms_stats.default_degree, estimate_percent=>
    dbms_stats.auto_sample_size, method_opt=>
    'for all columns size 1 for columns size 254 GID' , granularity=>'GLOBAL') ;
ALTER TABLE mv_name MINIMIZE RECORDS_PER_BLOCK ;
```

DBMS_ODM パッケージの使用

OLAP のデータ管理パッケージ DBMS_ODM のプロシージャは、ディメンションのマテリアライズド・ビューを作成するスクリプト、およびグルーピング・セット形式でファクトのマテリアライズド・ビューを作成するスクリプトを生成します。これらのスクリプトを、それぞれの形式で実行するか、スクリプトを実行する前に編集するか、または単に独自の SQL スクリプトを作成するモデルとして使用することができます。

重要： スクリプトを編集する場合は、DBMS_ODM で生成されるマテリアライズド・ビューと同じ構造のマテリアライズド・ビューが生成されるようにする必要があります。このようにしないと、生成されたマテリアライズド・ビューに OLAP API からアクセスできなくなります。

手順：グルーピング・セット形式のマテリアライズド・ビューの作成

次の手順を実行して、キューブのグルーピング・セット形式のマテリアライズド・ビューを作成します。

1. OLAP カタログにキューブを作成します。Enterprise Manager または CWM2 プロシージャを使用できます。CWM2 プロシージャを使用する場合は、必ずキューブをスター・スキーマにマップします。
2. UTL_FILE_DIR パラメータを有効なディレクトリに設定して、データベースがスクリプトをファイルに書き込めるようにします。
3. メタデータ所有者の識別情報を使用して、SQL*Plus にログインします。
4. 現在、キューブに存在するマテリアライズド・ビューを削除します。
5. ディメンションのマテリアライズド・ビューを生成するスクリプトを作成します。キューブのディメンションごとに DBMS.CREATEDIMMV_GS を実行します。
6. 次の3つの手順に従って、キューブのファクト表のグルーピング・セット形式のマテリアライズド・ビューを生成するスクリプトを作成します。
 - a. DBMS_ODM.CREATEDIMLEVTUPLE を実行して、SYS.OLAPTABLELEVELS 表を作成します。この表には、キューブのすべてのディメンションおよび各ディメンションのすべてのレベルが示されます。

デフォルトでは、すべてのディメンションのすべてのレベルがマテリアライズド・ビューに含める対象として選択されています。集計データを格納する必要がないとわかっているレベルがある場合は、表を編集して不要なレベルを選択解除します。
 - b. DBMS_ODM.CREATECUBELEVELTUPLE を実行して、SYS.OLAPTABLELEVELTUPLES 表を作成します。この表には、キューブのレベルの考え得るすべての組合せ（グルーピング・セット）が示されます。SYS.OLAPTABLELEVELS で選択したレベルを含むグルーピング・セットのみが、マテリアライズド・ビューに含める対象として選択されています。集計データを格納する必要がないとわかっているレベルの組合せがある場合は、表を編集して不要な組合せを選択解除します。
 - c. DBMS_ODM.CREATEFACTMV_GS を実行して、スクリプトを作成します。
7. オプションで、テキスト・エディタを使用してスクリプトを編集します。
8. 次のコマンドを使用して、SQL*Plus でスクリプトを実行します。

```
@/users/oracle/OraHome1/olap/mvscript.sql;
```

例 : Sales キューブのグルーピング・セット形式のマテリアライズド・ビューの作成

DRUG_DEPOT スキーマの DRUGSTORE キューブのマテリアライズド・ビューを作成するものとします。このキューブには、売上、費用、数量および予測のデータが格納されています。このキューブは、最低レベルのデータのみを格納するファクト表と、CHANNEL、GEOGRAPHY、PRODUCT および TIME のディメンション表にマップされています。各ディメンションが持つ階層は1つです。

1. まず、ディメンションのマテリアライズド・ビューを作成するスクリプトを生成します。次の文は、chanmv、prodmv、geogmv および timemv の各スクリプトを /dat1/scripts/drug_depot に作成します。

```
EXEC DBMS_ODM.CREATEDIMMV_GS
('drug_depot', 'channel','chanmv','/dat1/scripts/drug_depot');
EXEC DBMS_ODM.CREATEDIMMV_GS
('drug_depot', 'product','prodmv','/dat1/scripts/drug_depot');
EXEC DBMS_ODM.CREATEDIMMV_GS
('drug_depot', 'geography','geogmv','/dat1/scripts/drug_depot');
EXEC DBMS_ODM.CREATEDIMMV_GS
('drug_depot', 'time','timemv','/dat1/scripts/drug_depot');
```

2. これらのスクリプトを実行して、ディメンションのマテリアライズド・ビューを作成します。
3. 次に、ファクトのマテリアライズド・ビュー用に、ディメンションのレベルの表を作成します。

```
EXEC DBMS_ODM.CREATEDIMLEVTUPLE('drug_depot', 'drugstore');
```

レベルの表 SYS.OLAPTABLELEVELS は、このセッション専用の一時表です。次のコマンドで、この表の内容を表示できます。

```
select * from SYS.OLAPTABLELEVELS;
```

SCHEMA_NAME	DIMENSION_NAME	CUBE_NAME	LEVEL_NAME	SELECTED
-----	-----	-----	-----	-----
DRUG_DEPOT	CHANNEL	DRUGSTORE	TOTAL	1
DRUG_DEPOT	CHANNEL	DRUGSTORE	CHANNEL_CLASS	1
DRUG_DEPOT	CHANNEL	DRUGSTORE	CHANNEL_ID	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	TOTAL	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	REGION	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	SUB_REGION	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	COUNTRY	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	STATE_PROVINCE	1
DRUG_DEPOT	PRODUCT	DRUGSTORE	TOTAL	1
DRUG_DEPOT	PRODUCT	DRUGSTORE	PROD_CATEGORY	1
DRUG_DEPOT	PRODUCT	DRUGSTORE	PROD_SUBCATEGORY	1
DRUG_DEPOT	PRODUCT	DRUGSTORE	ID	1
DRUG_DEPOT	TIME	DRUGSTORE	Year	1
DRUG_DEPOT	TIME	DRUGSTORE	Quarter	1
DRUG_DEPOT	TIME	DRUGSTORE	Month	1

初期状態では、SYS.OLAPTABLELEVELS のすべてのレベルが選択されています (SELECTED 列の「1」)。

4. すべてのチャネルおよびすべての製品カテゴリにわたる、各地域およびサブ地域の集計データを格納するものとします。また、月レベルのデータは不要で、四半期レベルおよび年レベルのデータのみをマテリアライズド・ビューに格納します。

SYS.OLAPTABLELEVELS を編集して、TOTAL を除く CHANNEL のすべてのレベル、GEOGRAPHY の STATE_PROVINCE レベル、PRODUCT の PROD_SUBCATEGORY および個々の製品 ID、TIME の Month を選択解除します。

```
update SYS.OLAPTABLELEVELS set selected = 0
  where LEVEL_NAME in ('CHANNEL_ID','CHANNEL_CLASS', 'STATE_PROVINCE',
                      'ID','PROD_SUBCATEGORY','Month');
select * from sys.olaptablelevels;
```

SCHEMA_NAME	DIMENSION_NAME	CUBE_NAME	LEVEL_NAME	SELECTED
-----	-----	-----	-----	-----
DRUG_DEPOT	CHANNEL	DRUGSTORE	TOTAL	1
DRUG_DEPOT	CHANNEL	DRUGSTORE	CHANNEL_CLASS	0
DRUG_DEPOT	CHANNEL	DRUGSTORE	CHANNEL_ID	0
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	TOTAL	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	REGION	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	SUB_REGION	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	COUNTRY	1
DRUG_DEPOT	GEOGRAPHY	DRUGSTORE	STATE_PROVINCE	0
DRUG_DEPOT	PRODUCT	DRUGSTORE	TOTAL	1
DRUG_DEPOT	PRODUCT	DRUGSTORE	PROD_CATEGORY	1
DRUG_DEPOT	PRODUCT	DRUGSTORE	PROD_SUBCATEGORY	0
DRUG_DEPOT	PRODUCT	DRUGSTORE	ID	0
DRUG_DEPOT	TIME	DRUGSTORE	Year	1
DRUG_DEPOT	TIME	DRUGSTORE	Quarter	1
DRUG_DEPOT	TIME	DRUGSTORE	Month	0

5. 次に、SYS.OLAPTABLELEVELTUPLES 表を作成します。この表（これもセッション専用の一時的表）には、キューブのレベルの考え得るすべての組合せが格納されます。4 つのディメンションの各レベルの組合せ（グルーピング・セット）には、それぞれ識別番号が付けられています。SYS.OLAPTABLELEVELS で選択したレベルを含むグルーピング・セットの SELECTED 列には、「1」のマークが付けられています。

```
exec dbms_olap.createcubeleveltuple('drug_depot','drugstore');
select * from sys.olaptableleveltuples;
```

ID	SCHEMA_NAME	CUBE_NAME	DIMENSION_NAME	LEVEL_NAME	SELECTED
---	-----	-----	-----	-----	-----
1	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	STATE_PROVINCE	0
1	DRUG_DEPOT	DRUGSTORE	PRODUCT	ID	0
1	DRUG_DEPOT	DRUGSTORE	CHANNEL	CHANNEL_ID	0
1	DRUG_DEPOT	DRUGSTORE	TIME	Month	0
2	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	0
2	DRUG_DEPOT	DRUGSTORE	PRODUCT	ID	0

2	DRUG_DEPOT	DRUGSTORE	CHANNEL	CHANNEL_ID	0
2	DRUG_DEPOT	DRUGSTORE	TIME	Month	0
.					
.					
.					
112	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
112	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
112	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
112	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
113	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	SUB_REGION	1
113	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
113	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
113	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
.					
.					
.					
179	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1
179	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
179	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
179	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
180	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	TIME	Year	1

SYS.OLAPTABLELEVELTUPLES 表には 720 の行が格納されており、180 の一意なレベル・タブ
ル（グルーピング・セット）があります。180 というのは、キューブの各ディメンションの
レベル数をかけた結果です（3 × 5 × 4 × 3）。CHANNEL には 3 つのレベル、GEOGRAPHY に
は 5 つのレベル、PRODUCT には 4 つのレベル、TIME には 3 つのレベルがあります。

180 のグルーピング・セットのうち、マテリアライズド・ビューに含める対象として選択さ
れているのは 16 個のみです。次の文で、選択されている 64 の行（16 × 4）を表示できま
す。

```
select * from sys.olaptableleveltuples where SELECTED = 1;
```

ID	SCHEMA_NAME	CUBE_NAME	DIMENSION_NAME	LEVEL_NAME	SELECTED
--	-----	-----	-----	-----	-----
112	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
112	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
112	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
112	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
113	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	SUB_REGION	1
113	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
113	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
113	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
114	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1
114	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1

114	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
114	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
115	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
115	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
115	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
115	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
117	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
117	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
117	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
117	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
118	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	SUB_REGION	1
118	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
118	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
118	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
119	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1
119	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
119	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
119	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
120	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
120	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
120	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
120	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
172	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
172	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
172	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
172	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
173	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	SUB_REGION	1
173	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
173	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
173	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
174	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1
174	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
174	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
174	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
175	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
175	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
175	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
175	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
177	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
177	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
177	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
177	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
178	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	SUB_REGION	1
178	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
178	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
178	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
179	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1

179	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
179	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
179	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
180	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	TIME	Year	1

6. 各 SUB_REGION の Year 別の製品 TOTAL を格納するとします。また、SUB_REGION レベルを含むその他のグルーピング・セットの集計は格納しないものとします。

グルーピング・セットの 113、118、173 および 178 は、いずれも GEOGRAPHY の SUB_REGION レベルを使用しています。

ID	GEOGRAPHY	PRODUCT	CHANNEL	TIME
--	-----	-----	-----	-----
113	SUB_REGION	PROD_CATEGORY	TOTAL	Quarter
118	SUB_REGION	TOTAL	TOTAL	Quarter
173	SUB_REGION	PROD_CATEGORY	TOTAL	Year
178	SUB_REGION	TOTAL	TOTAL	Year

次のような文で、SYS.OLAPTABLELEVELTUPLES 表を編集できます。

```
update SYS.OLAPTABLELEVELTUPLES set selected = 0
      where ID in ('113','118', '173');
select * from sys.olaptableleveltuples where SELECTED = 1;
```

ID	SCHEMA_NAME	CUBE_NAME	DIMENSION_NAME	LEVEL_NAME	SELECTED
--	-----	-----	-----	-----	-----
112	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
112	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
112	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
112	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
114	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1
114	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
114	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
114	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
115	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
115	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
115	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
115	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
117	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
117	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
117	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
117	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
119	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1
119	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
119	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1

119	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
120	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
120	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
120	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
120	DRUG_DEPOT	DRUGSTORE	TIME	Quarter	1
172	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
172	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
172	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
172	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
174	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1
174	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
174	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
174	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
175	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
175	DRUG_DEPOT	DRUGSTORE	PRODUCT	PROD_CATEGORY	1
175	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
175	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
177	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	COUNTRY	1
177	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
177	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
177	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
178	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	SUB_REGION	1
178	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
178	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
178	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
179	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	REGION	1
179	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
179	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
179	DRUG_DEPOT	DRUGSTORE	TIME	Year	1
180	DRUG_DEPOT	DRUGSTORE	GEOGRAPHY	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	PRODUCT	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	CHANNEL	TOTAL	1
180	DRUG_DEPOT	DRUGSTORE	TIME	Year	1

7. ファクトのマテリアライズド・ビューを生成するスクリプトを作成するには、CREATEFACTMV_GS プロシージャを実行します。

```
exec dbms_odem.createfactmv_gs
('drug_depot','drugstore',
 'drugstore_mv','/dat1/scripts/drug_depot',TRUE);
```

スクリプト内の CREATE MATERIALIZED VIEW 文の GROUP BY GROUPING SETS 句には、次のグルーピング・セットが含まれます。

```
(TIMES.CALENDAR_YEAR, TIMES.CALENDAR_QUARTER, CHANNELS.TOTAL,
 PRODUCTS.TOTAL, PRODUCTS.PROD_CATEGORY, GEOGRAPHIES.TOTAL,
 GEOGRAPHIES.REGION, GEOGRAPHIES.SUB_REGION, GEOGRAPHIES.COUNTRY ),
(TIMES.CALENDAR_YEAR, TIMES.CALENDAR_QUARTER, CHANNELS.TOTAL,
```

```

PRODUCTS.TOTAL, PRODUCTS.PROD_CATEGORY, GEOGRAPHIES.TOTAL,
GEOGRAPHIES.REGION),
(TIMES.CALENDAR_YEAR, TIMES.CALENDAR_QUARTER, CHANNELS.TOTAL,
PRODUCTS.TOTAL, PRODUCTS.PROD_CATEGORY, GEOGRAPHIES.TOTAL),
(TIMES.CALENDAR_YEAR, TIMES.CALENDAR_QUARTER, CHANNELS.TOTAL,
PRODUCTS.TOTAL, GEOGRAPHIES.TOTAL, GEOGRAPHIES.REGION,
GEOGRAPHIES.SUB_REGION, GEOGRAPHIES.COUNTRY),
(TIMES.CALENDAR_YEAR, TIMES.CALENDAR_QUARTER, CHANNELS.TOTAL,
PRODUCTS.TOTAL, GEOGRAPHIES.TOTAL, GEOGRAPHIES.REGION),
(TIMES.CALENDAR_YEAR, TIMES.CALENDAR_QUARTER, CHANNELS.TOTAL,
PRODUCTS.TOTAL, GEOGRAPHIES.TOTAL),
(TIMES.CALENDAR_YEAR, CHANNELS.TOTAL, PRODUCTS.TOTAL,
PRODUCTS.PROD_CATEGORY, GEOGRAPHIES.TOTAL, GEOGRAPHIES.REGION,
GEOGRAPHIES.SUB_REGION, GEOGRAPHIES.COUNTRY),
(TIMES.CALENDAR_YEAR, CHANNELS.TOTAL, PRODUCTS.TOTAL,
PRODUCTS.PROD_CATEGORY, GEOGRAPHIES.TOTAL, GEOGRAPHIES.REGION),
(TIMES.CALENDAR_YEAR, CHANNELS.TOTAL, PRODUCTS.TOTAL,
PRODUCTS.PROD_CATEGORY, GEOGRAPHIES.TOTAL),
(TIMES.CALENDAR_YEAR, CHANNELS.TOTAL, PRODUCTS.TOTAL, GEOGRAPHIES.TOTAL,
GEOGRAPHIES.REGION, GEOGRAPHIES.SUB_REGION, GEOGRAPHIES.COUNTRY),
(TIMES.CALENDAR_YEAR, CHANNELS.TOTAL, PRODUCTS.TOTAL, GEOGRAPHIES.TOTAL,
GEOGRAPHIES.REGION, GEOGRAPHIES.SUB_REGION),
(TIMES.CALENDAR_YEAR, CHANNELS.TOTAL, PRODUCTS.TOTAL, GEOGRAPHIES.TOTAL,
GEOGRAPHIES.REGION),
(TIMES.CALENDAR_YEAR, CHANNELS.TOTAL, PRODUCTS.TOTAL, GEOGRAPHIES.TOTAL)

```

スクリプトの末尾にある次の文は、OLAP カタログのキューブに関連付けられた MV_SUMMARY_CODE を設定します。この設定は、このキューブに関連付けられたマテリアライズド・ビューがグルーピング・セット形式であることを示します。

```

execute cwm2_olap_cube.set_mv_summary_code
('DRUG_DEPOT', 'DRUGSTORE', 'GROUPINGSET') ;

```

8. drugstore_mv スクリプトを実行して、ファクトのマテリアライズド・ビューを作成します。

OLAP Server からのアップグレード

この付録では、Oracle OLAP Server リリース 6.3 から Oracle OLAP へのアップグレード方法、および両者の主な相違点について説明します。ここでは、このマニュアルの内容を理解するために役立つ参照情報を示します。

この付録では、次の項目について説明します。

- [管理](#)
- [アプリケーションのサポート](#)
- [プログラミング言語の変更](#)
- [Oracle OLAP Server データベースのスタンダード・フォームへの変換](#)

管理

Oracle OLAP は、Oracle Database Enterprise Edition のオプションとしてインストールされ、現在は Oracle Database に統合されています。OLAP Server はサービス環境で実行されますが、Oracle OLAP は Oracle Database のカーネルで実行されます。

Oracle では、**データベース**という用語はリレーショナル・データベースのことを表します。現在、OLAP Server データベースは[アナリティック・ワークスペース](#)と呼ばれます。Oracle OLAP では、アナリティック・ワークスペースを一時データ・キャッシュまたは永続データ・リポジトリとして使用できます。永続アナリティック・ワークスペースは、LOB としてリレーショナル表に格納され、これが表領域に格納されます。.db ファイルは存在しません。

Oracle OLAP の管理作業は、データベース・ツール・セットにマージされています。

管理ツール

Oracle Enterprise Manager には、Oracle Database を管理するツールが用意されており、すべてのプラットフォームで共通のユーザー・インタフェースが提供されます。他の Oracle Database のパフォーマンス統計と同様に、OLAP のパフォーマンス・データをシステム表で

集計できます。Oracle Enterprise Manager には、SQL へのグラフィカル・インタフェースが用意されています。

Express Instance Manager の oesmgr および oescmd は使用できません。

ユーザーの認証

Oracle OLAP は、Oracle Database をその識別情報でインストールしているユーザーの場合を除いて、オペレーティング・システムの識別情報を使用しません。他に目的がない場合は、OLAP Server (DBA ユーザー、初期化ユーザー、デフォルト・ユーザー、個々のユーザー名など) で使用するために作成されたその他のオペレーティング・システムの識別情報を削除できます。

認証は、すべて Oracle Database によって実行されます。アプリケーションは、セッションを開く前に、常に資格証明を提示する必要があります。また、これらの資格証明は、リレーショナル・データベースに格納されているユーザー名およびパスワードと一致する必要があります。ユーザーが Oracle OLAP にアクセスする前に、データベースでユーザー名とパスワードを定義する必要があります。

ユーザーがオペレーティング・システム・ファイルにアクセスするには、物理的なディレクトリ・パスにマップされているディレクトリ別名へのアクセス権が必要です。このアクセス権は、個々のユーザー ID またはデータベース・ロールに付与されます。

参照： OLAP 管理作業の詳細は、[第 12 章](#)を参照してください。

データ転送

Oracle OLAP セッションは、常にデータベースに接続されています。個別またはオプションの手順として、データベースへの接続はオープンしません。

次の方法で、アナリティック・ワークスペース・オブジェクト (変数、ディメンションなど) とリレーショナル表間でデータをコピーできます。

- PL/SQL パッケージの DBMS_AWM のプロシージャを使用すると、リレーショナル表からアナリティック・ワークスペースを作成できます。Analytic Workspace Manager では、このパッケージへのグラフィカル・インタフェースが用意されています。
- OLAP DML の SQL コマンドを使用すると、後で使用するために、ディメンションおよび変数にデータをフェッチできます。新しい SQL の IMPORT コマンドによって、リレーショナル表からアナリティック・ワークスペースへの大量のデータ転送が容易になります。また、新しい SQL の INSERT DIRECT コマンドによって、アナリティック・ワークスペースからリレーショナル表へのデータ転送が容易になります。
- SQL テーブル・ファンクションを使用すると、SQL ベースのアプリケーションでアナリティック・ワークスペースのデータを操作および抽出できます。OLAP Server では、データ転送を外部で開始することは許可されませんでした。Analytic Workspace Manager では、OLAP_TABLE ファンクションへのグラフィカル・インタフェースが用意されています。

ODBC は使用できないため、Oracle OLAP から直接サード・パーティのデータベースにアクセスすることはできません。

Oracle Express Relational Access Administrator および Oracle Express Relational Access Manager は使用不可です。

ローカライゼーション

OLAP Server の言語サポートは、Oracle グローバリゼーション・テクノロジーに置き換えられます。Oracle グローバリゼーション・テクノロジーによって、拡張されたローカライゼーション・サポートが提供され、OLAP Server のローカライゼーション機能を使用する場合よりも管理が容易になります。Oracle Database および Oracle OLAP は、インストール中に選択した同一のキャラクタ・セットを使用します。

変換表を使用していた OLAP Server データベースをアップグレードする場合、Oracle OLAP では変換表は不要なため、これらの変換表を削除できます。同様に、変換表をサポートしていた廃止コマンドおよびキーワードが OLAP Server プログラムで使用されていないことを確認する必要があります。

OLAP DML には、グローバリゼーション・テクノロジーのサポートが追加されています。これらのオプションによって、アプリケーションで、現行のローカライゼーション設定を問い合わせたり、デフォルト言語と地域によって制御される動作を変更できます。

表 A-1 に、OLAP Server のキャラクタ・セットと、それと同等の Oracle で使用可能な Unicode キャラクタ・セットを示します。OLAP Server データベースをインポートするか、Oracle OLAP を使用して、外部ファイルのマルチバイト・データにアクセスする場合、適切なデータベース・キャラクタ・セットの識別について、次の表を参照してください。OLAP Server CHARSET オプションは廃止されていることに注意してください。

表 A-1 マルチバイト・キャラクタ・セットの対応

OLAP Server	Unicode キャラクタ・セット
EUC	JA16EUC
SHIFTJIS	JA16SJIS
HANGEUL	KO16KSC5601
SCHINESE	ZHS16GBK
TCHINESE	ZHT16BIG5

アプリケーションのサポート

Oracle OLAP では、Java API または SQL を使用して、アプリケーションで直接多次元データにアクセスできます。Express SPL で書かれたプログラムは、これらのプログラムを使用

して実行できます。すべての SPL プログラムを再確認して、使用できないコマンドを削除し、新しい機能を使用してください。

Analytic Workspace Manager は、リレーショナル表から **データベース・スタンダード・フォーム** のアナリティック・ワークスペースを作成したり、データを集計したり、BI Beans、OLAP API または Oracle Discoverer によるワークスペースへのアクセスを有効化するウィザードを提供します。有効化には、アナリティック・ワークスペースに格納されているデータのリレーショナル・ビューの生成、およびこれらのビューに対するアプリケーションに適切なタイプのメタデータの作成が含まれます。

OLAP Server で使用するために開発された Windows C++、HTML または Java アプリケーションは実行できません。

参照： リレーショナル表からスタンダード・フォームのアナリティック・ワークスペースを作成する方法については、[第 6 章](#)を参照してください。

プログラミング環境

Oracle OLAP 用のアプリケーションは、OLAP API または BI Beans を使用して Java で開発できます。SQL ベースのアプリケーションでは、ビューを介して OLAP データにアクセスできます。また、OLAP_TABLE ファンクションを使用して、直接 OLAP データを操作できます。

OLAP Worksheet によって、OLAP DML または SQL でストアド・プロシージャを開発するための対話型の環境が提供されます。PL/SQL の DBMS_AW プロシージャを使用すると、SQL 環境から OLAP DML コマンドを実行できます。

Express Administrator、Personal Express または Express Connection Utility を使用して、Oracle OLAP に接続することはできません。

参照：

- OLAP API および BI Beans については、[第 4 章](#)を参照してください。
- OLAP DML コマンドの実行方法については、[第 9 章](#)を参照してください。

通信

Oracle OLAP によって、Oracle Call Interface (OCI) および Java Database Connectivity (JDBC) での通信が提供されます。

OLAP Worksheet は、アナリティック・ワークスペースとの通信に XCA を使用します。ただし、XCA はユーザー開発アプリケーションではサポートされておらず、予期しない結果が発生する可能性があります。

SNAPI は使用できません。セッション共有はサポートされていません。

メタデータ

OLAP API および BI Beans を使用すると、アナリティック・ワークスペースまたはリレーショナル表に格納されているデータを問い合わせることができます。データベース管理者は、両方のデータソースのタイプの OLAP カタログ・メタデータを定義します。メタデータは、表およびビューに格納されます。

データベース・スタンダード・フォームは、Analytic Workspace Manager のサーバー・ツールで使用するための、アナリティック・ワークスペースに格納されているメタデータのタイプです。メタデータは、ワークスペース・オブジェクトのプロパティおよびカタログに格納されます。これらは、特別なディメンション、変数および値セットとして実装されます。

Oracle Express Administrator は、Oracle OLAP では使用できません。また、Oracle Express Administrator によって生成された Oracle Express Objects のメタデータは、OLAP API または BI Beans では使用されません。

参照：

- OLAP カタログについては、[第 5 章](#)を参照してください。
- データベース・スタンダード・フォームについては、[第 8 章](#)を参照してください。

プログラミング言語の変更

Express SPL（現在は、OLAP データ操作言語または OLAP DML と呼ばれています）には、多くの変更が追加されました。

新しいコマンド

OLAP DML には、次の領域のコマンドのサポートが追加されています。

アロケーション

動的モデルの実行

アナリティック・ワークスペースおよびリレーショナル・データベース間の大量のデータ転送

バイト操作ファンクション

日付変換ファンクション

新しいデータ型

廃止コマンド

次の機能のサポートは削除されました。

EXTCALL

ODBC

SNAPI

XCA

オペレーティング・システムのコマンド

結合ディメンションおよび ROLLUP コマンドは現在も使用できますが、管理が簡単でパフォーマンスが優れている点から、コンポジット・ディメンションおよび **aggmap** をかわりに使用することを強くお勧めします。

UPDATE および COMMIT

UPDATE コマンドは、一時表領域から永続表領域にアナリティック・ワークスペースの変更を移動します。Oracle OLAP セッションまたは SQL から COMMIT コマンドを実行するまで、変更は永続的に保存されません。COMMIT によって、永続表領域がディスクに書き込まれます。

永続表領域に移動されていない変更は、コミットされません。最初にアナリティック・ワークスペースを更新せずに COMMIT を発行した場合、前回 UPDATE を発行した後に加えたアナリティック・ワークスペースの変更はディスクにコミットされません。

COMMIT コマンドでは、SQL の COMMIT コマンドを実行します。セッション中に Oracle OLAP またはその他のアクセス方法（SQL など）で行われたデータベースへのすべての変更がコミットされます。

Oracle OLAP Server データベースのスタンダード・フォームへの変換

EIF ファイルは、あるデータベースから別のデータベースにアナリティック・ワークスペースの内容を転送して、OLAP Server データベースからアップグレードするために使用します。EIF ファイルを使用してオブジェクトを転送するだけで、OLAP Server データベースからアナリティック・ワークスペースを作成できます。

作業としては、データベース・スタンダード・フォームのアナリティック・ワークスペースを作成の方が複雑です。したがって、現行の世代の Oracle OLAP ツールを使用できます。また、OLAP Server メタデータを使用して、スタンダード・フォーム・メタデータを作成できる場合があります。それ以外の場合は、新しい論理メタデータ・モデルを定義する必要があります。

CREATE_DB_STDFORM の使用対象者

OLAP Server データベースに、Oracle Express Administrator によって作成された Oracle Express Objects のメタデータが含まれている場合、変換プログラムの **CREATE_DB_STDFORM** を使用できます。Oracle Express Objects のメタデータがない場合、OLAP ツールが機能するスタンダード・フォームのメタデータは、**CREATE_DB_STDFORM** で生成できません。

特にソース・データがフラット・ファイルにある場合は、可能なかぎり CREATE_DB_STDFORM を使用します。現時点では、フラット・ファイルからスタンダード・フォームのアナリティック・ワークスペースを直接作成できるツールはありません。

ソース・データが表またはビューにある場合、CREATE_DB_STDFORM を使用して OLAP Server データベースを変換するか、他のツールを使用してソース・データから直接アナリティック・ワークスペースを作成するかを選択できます。CREATE_DB_STDFORM を使用すると、OLAP カタログの論理モデルを再定義せずに、Oracle Express Objects メタデータを使用できます。ただし、他の手順は手動で実行する必要があります（A-7 ページの「CREATE_DB_STDFORM では提供されない機能」を参照）。要件に応じて最適の方法を選択できます。

表 A-2 に、アップグレードのオプションを示します。

表 A-2 OLAP Server データベースのアップグレード方法の選択

Oracle Express Objects メタデータの有無	ソース・データの場合	スタンダード・フォームのアナリティック・ワークスペースの作成手段
有	表またはビュー	CREATE_DB_STDFORM または第 6 章で示されているいずれかの方法
有	フラット・ファイル	CREATE_DB_STDFORM
無	表またはビュー	第 6 章で示されているいずれかの方法
無	フラット・ファイル	Oracle Warehouse Builder（第 6 章を参照）または第 11 章で示されている方法

CREATE_DB_STDFORM で提供される機能

CREATE_DB_STDFORM を使用すると、データに対して短時間で BI Beans の使用を開始できます。Oracle Express Objects メタデータからデータベース・スタンダード・フォームのメタデータへの変換手順では、プログラムを 1 つ実行します。次に、Analytic Workspace Manager のダイアログ・ボックスを使用して、OLAP API および BI Beans に対してアナリティック・ワークスペースを有効化できます。EIF ファイルのインポートから BI Beans アプリケーションを使用したアナリティック・ワークスペースのビューの問合せに至るプロセス全体が、短時間で済み、完全に自動化されています。

ベース・レベルのデータのみをロードする場合、Analytic Workspace Manager の集計ウィザードを使用すると、集計プランを作成およびデプロイできます。この集計方法の場合、ROLLUP コマンドを使用するより処理が速く、柔軟性が高くなります。

CREATE_DB_STDFORM では提供されない機能

変換プロセスでは、アナリティック・ワークスペースを作成する通常の最初の手順（OLAP カタログにおける論理データ・モデルの開発およびデータソースへの論理オブジェクトのマップ）は実行する必要がありません。ただし、論理モデルを変更する場合は、OLAP カタログを変更します。その場合、ツールで OLAP カタログの変更をリフレッシュして、適切な

変更をスタンダード・フォームのカタログに適用します。このメンテナンス・プロセスは、`CREATE_DB_STDFORM` で変換されたアナリティック・ワークスペースには使用できません。そのため、次の作業を手動で実行する必要があります。

- データに対して時間ベースの分析を実行する場合、**Time** ディメンションの終了日属性および期間属性を移入する必要があります。この付録のサンプル・プログラムを参考にしてください。
- アナリティック・ワークスペースは、廃止コマンドへの参照が記述されたプログラムを含んでいる場合があります。これは変更する必要があります。また、新機能のいくつかを使用する場合もあります。たとえば、スパースなデータを（結合ではなく）コンボジットで処理できます（処理していない場合）。この変更を行うには、新しい変数を定義し、古い変数のデータをコピー（またはデータソースからデータを再ロード）します。
- 変換済のアナリティック・ワークスペースに新しいデータをコピーする場合、**Analytic Workspace Manager** のリフレッシュ・ウィザードは使用できません。この場合、ロード・プログラムを変更するか、新規作成するか、手動で実行する必要があります。
- スタンダード・フォームのメタデータへの変更は、**MAINTAIN** コマンドおよび修飾データ参照を使用して、手動で適用する必要があります。

Oracle Express Objects メタデータからの変換

Oracle Express Objects 変換ツールは、アナリティック・ワークスペース上で機能します。このツールでは、Oracle Express Objects メタデータを使用して様々なオブジェクトのロールを識別し、次の処理を行います。

- 適切なスタンダード・フォームのプロパティを既存のオブジェクトを移入します。たとえば、Oracle Express Objects の言語ディメンションは、`ALL_LANGUAGES` の `AW$ROLE` 値で指定します。
- スタンダード・フォームに必要なディメンションおよびプロパティを持つスタンダード・フォーム・オブジェクトを新規作成し、それに既存のオブジェクトのデータをコピーします。たとえば、*hierdim* ディメンションでスタンダード・フォームの属性はディメンション化されますが、Oracle Express Objects の属性はディメンション化されません。XADEMO アナリティック・ワークスペースの場合、変換ツールによって、`CHANNEL`、`C0.HIERDIM` および `_XA_LANGDIM` でディメンション化された `CHANNEL_LONG_DESCRIPTION` 変数が作成され、それに `C0.LONGLABEL` の値が移入されます。
- スタンダード・フォームのカタログ、*member_gid* 変数、*member_inhier* 変数、*member_familyrel* リレーション、*member_levelrel* リレーションなどのスタンダード・フォームのメタデータ・オブジェクトを作成および移入します。これらのスタンダード・フォーム・オブジェクトの詳細は、[付録 C](#) を参照してください。

変換ツールでは、スタンダード・フォームのオブジェクトおよびプロパティを追加しますが、Oracle Express Objects のオブジェクトまたはプロパティは削除しません。これらは、必要に応じて手動で削除できます。

OLAP API および BI Beans では、時間ベースの分析をサポートするために、期間終了日属性および期間属性を持つ、レベルでソートされた Time ディメンションが必要です。

CREATE_DB_STDFORM の構文

CREATE_DB_STDFORM プログラムは、Oracle Express Objects 変換ツールを実行します。次に、このプログラムの構文を示します。

```
CREATE_DB_STDFORM(aw, [mode], [debug], [directory], [filename], [metacheck])
```

各項目の意味は次のとおりです。

aw はアナリティック・ワークスペースの名前 (TEXT 型) です。

mode はアタッチ・モード (RO | RW | RWX) です。

debug はデバッグを実行するかどうかを制御します (YES | NO)。

directory はデバッグ・ファイルが書き込まれるデータベース・ディレクトリ (TEXT 型) です。

filename はデバッグ・ファイルの名前 (TEXT 型) です。

metacheck は変換前にメタデータをチェックするかどうかを制御します (YES | NO)。

たとえば、次のコマンドは、読み込み / 書き込みモードで XADEMO をアタッチし、Oracle Express Objects メタデータに問題がないかを確認し、アナリティック・ワークスペースをスタンダード・フォームに変換し、画面にステータス・メッセージを表示します。

```
CALL CREATE_DB_STDFORM('xademo')
```

次のコマンドは、読み込み / 書き込み排他モードで XADEMO をアタッチし、データベース・ディレクトリ *xademo_dir* にあるファイル *xademo.log* にステータス・メッセージをリダイレクトします。また、メタデータのチェックも実行します。

```
CALL CREATE_DB_STDFORM('xademo', 'rwx', yes, 'xademo_dir', 'xademo.log')
```

手順 : Oracle Express Objects からスタンダード・フォームへの変換

新しいアナリティック・ワークスペースの作成や EIF ファイルのインポートなど、スタンダード・フォームへの変換手順の多くは、Analytic Workspace Manager のオブジェクト・ビューを使用して実行できます。ただし、次の手順では、OLAP DML コマンドに習熟しているユーザーが変換を実行すると想定して、OLAP Worksheet のコマンドライン・インタフェースを使用します。

Oracle Express Objects メタデータ変換ツールを使用して、スタンダード・フォームのアナリティック・ワークスペースを作成するには、次の手順を実行します。

1. Oracle Express Objects データベースから EIF ファイルを作成し、データベース・ディレクトリにマップされている物理ディレクトリにそのファイルをコピーします。

データベース・ディレクトリについては、12-9 ページの「[外部ファイルへのアクセスの許可](#)」を参照してください。

2. Analytic Workspace Manager を開き、Oracle Database にアタッチします (6-4 ページの「[Analytic Workspace Manager の概要](#)」を参照)。
3. 「Tools」メニューから、「OLAP Worksheet」を選択します。

別のウィンドウで OLAP Worksheet が開きます。OLAP Worksheet の使用方法については、9-4 ページの「[OLAP Worksheet を使用した OLAP DML の実行](#)」を参照してください。

4. 次のコマンドを使用して、EIF ファイルから新しいアナリティック・ワークスペースを作成します。

```
AW CREATE aw
IMPORT ALL FROM EIF FILE 'directory/filename.eif' DATA DFNS
UPDATE
COMMIT
```

5. 次のコマンドを使用して変換ツールを実行します。

```
CALL CREATE_DB_STDFORM('aw')
```

構文の説明については、A-9 ページの「[CREATE_DB_STDFORM の構文](#)」を参照してください。

6. 変換ツールの実行が正常に完了した後、変更を保存します。

```
UPDATE
COMMIT
```

これで、スタンダード・フォームのアナリティック・ワークスペースが作成されました。

7. BI Beans に対してワークスペースを有効化します。6-23 ページの「[アプリケーションに対するアナリティック・ワークスペースの有効化](#)」を参照してください。

この手順は、変換手順の他の手順が完了した後も実行できます。

8. 時間ベースの分析の場合、終了日属性と期間属性を移入します。

9. 新しいデータでアナリティック・ワークスペースをリフレッシュするには、データ・ローダー・プログラムを変更および実行します (A-11 ページの「[ロード・プログラムの変更](#)」を参照)。

Time 属性の移入

スタンダード・フォームの **Time** ディメンションには、次の特性があります。

- **AW\$TYPE** プロパティは、'Time' の値を持つ。
- ディメンション・メンバーはレベル内で時系列順にソートされている。
- 期間の終了日属性および期間属性が定義および移入されている。

変換プロセスでは、**AW\$TYPE** プロパティを設定し、終了日と期間についてスタンダード・フォームの属性を定義し、この情報をスタンダード・フォームのカタログに登録します。**Time** ディメンション・メンバーの順序は変更せず、属性も移入しません。

Time メンバーがレベル内で時系列順にソートされていない場合は、11-14 ページの「[ディメンション・メンバーのソート](#)」に示すようなプログラムを使用して正しくソートします。ここでは、アナリティック・ワークスペースに期間の[埋込み合計](#)ディメンションが含まれていると想定します。

終了日属性および期間属性の移入方法は、データソースおよび **Time** ディメンション・メンバーの形式によって異なります。元のデータソースから情報を取得できる場合（つまり、ソースが OLAP Server データベースの移入元である場合）、11-3 ページの「[フラット・ファイルの読み込み](#)」で示されているファイル読み込みプログラムを使用して情報をロードできます。それ以外の場合は、ディメンション・メンバーまたはその記述から情報を導出する必要があります。この方法の例については、A-14 ページの「[XADEMO Time 属性の移入](#)」を参照してください。

ロード・プログラムの変更

Analytic Workspace Manager のリフレッシュ・ウィザードでは、DBMS_AWM プロシージャで作成されたアナリティック・ワークスペースのみを操作します（[第 6 章](#)を参照）。

CREATE_DB_STDFORM を使用してアナリティック・ワークスペースを作成する場合、新しいデータの取得に必要な情報をリフレッシュ・ウィザードで提供するメカニズムは不要です。OLAP DML プログラムを使用して手動でアナリティック・ワークスペースをリフレッシュする必要があります。

アナリティック・ワークスペースには、OLAP Server データベースをリフレッシュするために Express Administrator で生成されたプログラムが含まれている場合があります。このプログラムは、現行の状態では使用できないため、アナリティック・ワークスペースで使用できるよう変更します。

ロード・プログラムから次のコードを削除します。

- EDDE.MSG へのコール。これは、管理用 GUI に OLAP Server エラー・メッセージを表示するプログラムでした。このプログラムへのコールを削除しても、プログラムの操作には影響しません。
- EDDE.HIERMNT へのコール。これは、ディメンション階層に関連付けられたメタデータを管理するプログラムでした。現在は、アナリティック・ワークスペースでは使用不可

になっており、XPDDDATA データベースに格納されたデータの情報も利用できません。スタンダード・フォーム・メタデータの変更は手動で管理する必要があります。

- Oracle Database との接続を確立するコード。アナリティック・ワークスペースは Oracle Database の一部であるため、接続は常に開いています。

ロード・プログラムでは、ディメンションおよびメジャーのみをリフレッシュし、ディメンション属性、階層オブジェクトおよびレベル・オブジェクト、またはスタンダード・フォームのカatalogはリフレッシュしません。ディメンションに関連付けられたスタンダード・フォーム・オブジェクト用のロード・プログラムの記述については、[第 11 章](#)を参照してください。スタンダード・フォームのカatalogについては、[付録 C](#)を参照してください。

例：スタンダード・フォームへの XADEMO データベースの変換

この例では、OLAP Server データベース XADEMO のオブジェクトおよび Oracle Express Objects メタデータを含む EIF ファイルを使用します。OLAP Server データベースの変換では、XADEMO を理解しているものと想定します。

スタンダード・フォームの XADEMO アナリティック・ワークスペースの作成

xademo.eif という EIF ファイルが、システム・ディレクトリ %users%\oracle\xademo_files に配置されているとします。このファイルからスタンダード・フォームのアナリティック・ワークスペースを作成するには、次の手順を実行します。

1. SYSTEM ユーザーとして Oracle Database にログインし、XADEMO ユーザー、永続表領域、一時表領域および EIF ファイルへのアクセス用データベース・ディレクトリを作成します。

```
CREATE TABLESPACE olapdata DATAFILE '$ORACLE_HOME/oradata/olapdata.dbf'
    SIZE 5M REUSE AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED
    EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

CREATE TEMPORARY TABLESPACE olaptmp TEMPFILE
    '$ORACLE_HOME/oradata/olaptmp.tmp'
    SIZE 5M REUSE AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;

CREATE USER xademo IDENTIFIED BY 'xademo'
    DEFAULT TABLESPACE olapdata
    TEMPORARY TABLESPACE olaptmp
    QUOTA UNLIMITED ON olapdata
    QUOTA UNLIMITED ON olaptmp
    ACCOUNT UNLOCK;

CREATE DIRECTORY xademo_dir as '/users/oracle/OraHome1/xademo_files';
```

```
GRANT READ ON DIRECTORY xademo_dir TO xademo;
```

この作業の詳細は、[第 12 章](#)を参照してください。

- 2. Analytic Workspace Manager を開き、XADEMO ユーザーとして Oracle Database に接続します。
- 3. OLAP Worksheet を開きます。
- 4. 次のように、EIF ファイルからアナリティック・ワークスペースを作成します。

```
AW CREATE xademo
IMPORT ALL FROM EIF FILE 'olapdata/xademo.EIF' DATA DFNS
UPDATE
COMMIT
```

- 5. 次のように、アナリティック・ワークスペースをデータベース・スタンダード・フォームに変換します。

```
CALL CREATE_DB_STDFORM('xademo')
UPDATE
COMMIT
```

- 6. TIME_END_DATE 変数および TIME_TIME_SPAN 変数を移入します (次の項を参照)。
XADEMO データベースにデータ・ローダー・プログラムがないため、ここでは例は示しません。ファイル・ロード・プログラムおよび SQL フェッチ・プログラムの例については、[第 11 章](#)を参照してください。

XADEMO の Time ディメンションについて

TIME ディメンションには階層が 2 つあり、T0.LEVELDIM ディメンションに示されています。これらの階層には、STANDARD および YTD という名前が付いています。次のレポートに、各レベルの TIME メンバーのサンプルを示します。

```
REPORT DOWN time t0.levelrel W 20 t0.lvllabfrm
```

TIME	-----T0.HIERDIM-----			
	-----STANDARD-----		-----YTD-----	
	T0.LEVELREL	T0.LVLLABFRM	T0.LEVELREL	T0.LVLLABFRM
JAN96	L3	Month(s)	L5	YTD Month(s) Detail
FEB96	L3	Month(s)	L5	YTD Month(s) Detail
Q1.96	L2	Quarter(s)	NA	NA
LAST.YTD	NA	NA	L4	YTD Summaries
1996	L1	Year(s)	NA	NA

XADEMO Time 属性の移入

例 A-1 に示す POP_TIME_ATTRS プログラムは、TIME_END_DATE 変数および TIME_TIME_SPAN 変数を移入します。

TIME_END_DATE については、プログラムは ENDDATE ファンクションを使用して、各期間の最終日を識別します。ENDDATE ファンクションは、データ型が時間（MONTH および YEAR など）のディメンションのみを操作します。ただし、XADEMO TIME ディメンションのデータ型は TEXT です。したがって ENDDATE ファンクションを使用するには、複数の変換が必要です。プログラムでは次の手順を実行します。

1. レベルごとに、適切なデータ型（MONTH、QUARTER または YEAR）でディメンションを定義します。例ではディメンションに、M_TEMP、Q_TEMP および Y_TEMP という名前が付いています。
2. 特定のレベルのディメンション・メンバーの名前を値セットに格納します。例では値セットに T_LIST という名前が付いています。
3. T_LIST 値セットの現行のステータスを使用して、メンバーを新しいディメンション（M_TEMP、Q_TEMP および Y_TEMP）に追加します。

TIME_TIME_SPAN については、プログラムは、30APR96 などの値を含む TIME_END_DATE から、月レベルで最初の 2 文字を抽出し、各月の日数を取得します。

次にプログラムは、ROLLUP コマンドを使用して、各四半期および各年の日数を計算します。T0.PARENT は、ディメンション・メンバー間の親子関係を識別するセルフ・リレーションです。ただし、T0.PARENT および TIME_TIME_SPAN は、いずれも T0.HIERDIM でディメンション化されているため、ROLLUP では T0.PARENT を使用できません。この場合、TIME のみでディメンション化した、リレーション TIME_PARENTREL をプログラムで作成し、それを T0.PARENT から移入して、ROLLUP コマンドで新しいリレーションを使用します。

効率性は、ROLLUP より aggmap の方が高くなります。ただし、この場合はディメンションが 1 つのみで、そのディメンションに集計値がすべて格納されているため、ROLLUP の方が利便性が少し高く、パフォーマンスの違いもごくわずかです。

例 A-1 TIME 属性を移入する OLAP DML プログラム

```
DEFINE POP_TIME_ATTRS PROGRAM
PROGRAM
VARIABLE _ytd TEXT           " Stores YTD time members
TRAP ON cleanup             " Divert processing on error to CLEANUP label

" Define dimensions for each level with date data types
IF NOT EXISTS('m_temp')
  THEN DEFINE m_temp DIMENSION MONTH
  ELSE MAINTAIN m_temp DELETE ALL

IF NOT EXISTS('q_temp')
  THEN DEFINE q_temp DIMENSION QUARTER
```

```

ELSE MAINTAIN q_temp DELETE ALL

" Format years like TIME year members (1997 instead of YR97)
IF NOT EXISTS('y_temp')
THEN DO
  DEFINE y_temp DIMENSION YEAR
  CONSIDER y_temp
  VNF <YYYY>
DOEND
ELSE MAINTAIN y_temp DELETE ALL

" Define a valueset to store time members
IF NOT EXISTS('t_list')
  THEN DEFINE t_list VALUESET TIME
  ELSE LIMIT t_list TO NA

" Define a one-dimensional time self-relation for rollup
IF NOT EXISTS('time_parentrel')
  THEN DEFINE time_parentrel RELATION time <time>
  ELSE time_parentrel = NA

" Initialize target variables
ALLSTAT
time_time_span = NA
time_end_date = NA
" *****
"   Set values for the STANDARD hierarchy
" *****
LIMIT t0.hierdim TO 'STANDARD'
" Select all time members at the month level
LIMIT time TO t0.levelrel EQ 'L3'
" Store months in the valueset
LIMIT t_list TO time
" Populate M_TEMP so all months have a MONTH data type
MAINTAIN m_temp MERGE values(t_list)
" Calculate the end date
FOR m_temp
  time_end_date(time, m_temp) = ENDDATE(m_temp)
" Extract the number of days in each month
time_time_span = CONVERT(EXTCHARS(time_end_date, 1, 2), DECIMAL)

" Store quarters in q_temp
LIMIT time TO t0.levelrel EQ 'L2'
LIMIT t_list TO time
MAINTAIN q_temp MERGE VALUES(t_list)
FOR q_temp
  time_end_date(time, q_temp) = ENDDATE(q_temp)

```

```
" Store years in y_temp
LIMIT time TO t0.levelrel EQ 'L1'
LIMIT t_list TO time
MAINTAIN y_temp MERGE VALUES(t_list)
FOR y_temp
    time_end_date(time, y_temp) = ENDDATE(y_temp)
" *****
"      Set values for the YTD hierarchy
" *****
LIMIT t0.hierdim TO 'YTD'
" Limit status of months to YTD
LIMIT time TO t0.levelrel EQ 'L5'
LIMIT t_list TO time
LIMIT m_temp TO t_list

" Calculate end date and time span for months
FOR m_temp
    time_end_date(time, m_temp) = ENDDATE(m_temp)
time_time_span = CONVERT(EXTCHARS(time_end_date, 1, 2), DECIMAL)

" Get current and previous YTD
LIMIT time TO t0.parent EQ 'LAST.YTD'
LIMIT time KEEP LAST 1
_ytd = time
time_end_date(time, 'LAST.YTD') = time_end_date(time, _ytd)
LIMIT time TO t0.parent EQ 'CURRENT.YTD'
LIMIT time KEEP LAST 1
_ytd = time
time_end_date(time, 'CURRENT.YTD') = time_end_date(time, _ytd)

" Rollup time span for quarters and years
LIMIT t0.hierdim TO ALL
LIMIT time TO ALL
FOR t0.hierdim
    DO
        time_parentrel = t0.parent
        ROLLUP time_time_span OVER time USING time_parentrel
    DOEND

CLEANUP:
" Delete temporary objects
DELETE m_temp q_temp y_temp t_list time_parentrel
END
```

GLOBALX の作成用プログラム

GLOBALX は、スター・スキーマ以外のソースから手動で移入したアナリティック・ワークスペースです（第 11 章「他のソースからのデータの取得」を参照）。この付録では、該当する章の例のレプリケートに必要であり、対応する項の説明とは異なる追加のソース・コードを示します。

この付録では、次の項目について説明します。

- ユーザーおよび表領域の定義用の SQL スクリプト
- GLOBALX スター・スキーマ用の SQL スクリプト
- OLAP カタログ・メタデータ用の SQL スクリプト

ユーザーおよび表領域の定義用の SQL スクリプト

最初に、GLOBALX 表領域を作成します。これは、GLOBALX ユーザーと GLOBALX_AW ユーザーの両方でデフォルトの表領域になるためです。次に、GLOBALX ユーザーを作成します。GLOBALX_AW ユーザーを作成する前に、スター・スキーマを定義します。

例 B-1 GLOBALX 表領域の作成用スクリプト

```
CREATE TABLESPACE globalx LOGGING
  DATAFILE '/users/oracle/global_data/globalx.dbf'
  SIZE 5M REUSE AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED
  EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

CREATE TEMPORARY TABLESPACE glotemp
  TEMPFILE '/users/oracle/global_data/glotemp.tmp'
  SIZE 5M REUNE AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
```

例 B-2 GLOBALX ユーザーの作成用スクリプト

```
/* Create the user and grant privileges */
```

```
CREATE USER globalx IDENTIFIED BY "globalx"
  DEFAULT TABLESPACE globalx
  TEMPORARY TABLESPACE glotemp
  QUOTA UNLIMITED ON globalx
  QUOTA UNLIMITED ON glotemp
  ACCOUNT UNLOCK;
GRANT CONNECT TO globalx;
GRANT OLAP_USER TO globalx;
GRANT CREATE ANY TYPE TO globalx;
GRANT CREATE ANY DIRECTORY TO globalx;

/* Create a database directory*/
CREATE OR REPLACE DIRECTORY gx AS '/users/oracle/globalx_files';
GRANT ALL ON DIRECTORY gx TO PUBLIC;
```

例 B-3 GLOBALX_AW ユーザーの作成用スクリプト

```
/* Create the user and grant privileges */
CREATE USER globalx_aw IDENTIFIED BY globalx_aw
  DEFAULT TABLESPACE globalx
  TEMPORARY TABLESPACE glotemp
  QUOTA UNLIMITED ON globalx
  QUOTA UNLIMITED ON glotemp
  ACCOUNT UNLOCK;
GRANT CONNECT TO globalx_aw;
GRANT OLAP_USER TO globalx_aw;
GRANT CREATE ANY TYPE TO globalx_aw;
GRANT CREATE ANY DIRECTORY TO globalx_aw;

/* Grant access to GLOBALX star schema */
GRANT SELECT ON GLOBALX.CHANNEL_DUMMY to GLOBALX_AW;
GRANT SELECT ON GLOBALX.PROD_DUMMY to GLOBALX_AW;
GRANT SELECT ON GLOBALX.CUSTOM_DUMMY to GLOBALX_AW;
GRANT SELECT ON GLOBALX.TIME_DUMMY to GLOBALX_AW;
GRANT SELECT ON GLOBALX.UNITS_DUMMY to GLOBALX_AW;
GRANT SELECT ON GLOBALX.PRICE_AND_COST_DUMMY to GLOBALX_AW;
```

GLOBALX スター・スキーマ用の SQL スクリプト

例 B-4 に、空の GLOBALX 表を作成するための SQL スクリプトを示します。

例 B-4 空の GLOBALX 表の作成用スクリプト

```
CREATE TABLE GLOBALX.CHANNEL_DUMMY
(CHANNEL_ID          NUMBER(5),
 CHANNEL_DSC         VARCHAR2(15),
```



```
ALL_CHANNELS_ID      NUMBER(5),
ALL_CHANNELS_DSC      VARCHAR2(15),
CONSTRAINT UK_ON_CHANNEL_ID PRIMARY KEY (CHANNEL_ID));
```

```
CREATE TABLE GLOBALX.CUSTOM_DUMMY
(SHIP_TO_ID          NUMBER(5),
SHIP_TO_DSC          VARCHAR2(30),
ACCOUNT_ID           NUMBER(5),
ACCOUNT_DSC           VARCHAR2(30),
MARKET_SEGMENT_ID    NUMBER(5),
MARKET_SEGMENT_DSC    VARCHAR2(15),
TOTAL_MARKET_ID       NUMBER(5),
TOTAL_MARKET_DSC      VARCHAR2(15),
WAREHOUSE_ID          NUMBER(5),
WAREHOUSE_DSC         VARCHAR2(15),
REGION_ID            NUMBER(5),
REGION_DSC            VARCHAR(15),
ALL_CUSTOMERS_ID      NUMBER(5),
ALL_CUSTOMERS_DSC     VARCHAR2(15),
CONSTRAINT UK_ON_SHIP_TO_ID PRIMARY KEY (SHIP_TO_ID));
```

```
CREATE TABLE GLOBALX.PROD_DUMMY
(ITEM_ID             NUMBER(5),
ITEM_DSC             VARCHAR2(31),
ITEM_PACKAGE_ID      VARCHAR2(20),
FAMILY_ID            NUMBER(5),
FAMILY_DSC           VARCHAR2(20),
CLASS_ID             NUMBER(5),
CLASS_DSC            VARCHAR2(15),
TOTAL_PRODUCT_ID      NUMBER(5),
TOTAL_PRODUCT_DSC     VARCHAR2(15),
CONSTRAINT UK_ON_ITEM_ID PRIMARY KEY (ITEM_ID));
```

```
CREATE TABLE GLOBALX.TIME_DUMMY
(MONTH_ID            NUMBER(5),
MONTH_DSC            VARCHAR2(10),
QUARTER_ID           NUMBER(5),
QUARTER_DSC          VARCHAR2(5),
YEAR_ID              NUMBER(5),
YEAR_DSC             VARCHAR2(5),
MONTH_TIMESPAN        NUMBER(5),
QUARTER_TIMESPAN      NUMBER(5),
YEAR_TIMESPAN         NUMBER(5),
MONTH_END_DATE        DATE,
QUARTER_END_DATE      DATE,
YEAR_END_DATE         DATE,
CONSTRAINT UK_ON_MONTH_ID PRIMARY KEY (MONTH_ID));
```

```
CREATE TABLE GLOBALX.PRICE_AND_COST_DUMMY
(ITEM_ID          NUMBER(5),
 MONTH_ID         NUMBER(5),
 UNIT_PRICE       NUMBER,
 UNIT_COST        NUMBER,
 CONSTRAINT FK_ON_ITEM_ID_01 FOREIGN KEY (ITEM_ID)
 REFERENCES PROD_DUMMY(ITEM_ID),
 CONSTRAINT FK_ON_MONTH_ID_01 FOREIGN KEY (MONTH_ID)
 REFERENCES TIME_DUMMY(MONTH_ID));

CREATE TABLE GLOBALX.UNITS_DUMMY
(CHANNEL_ID       NUMBER(5),
 ITEM_ID          NUMBER(5),
 SHIP_TO_ID       NUMBER(5),
 MONTH_ID         NUMBER(5),
 UNITS            NUMBER,
 CONSTRAINT FK_ON_CHANNEL_ID_02 FOREIGN KEY (CHANNEL_ID)
 REFERENCES CHANNEL_DUMMY(CHANNEL_ID),
 CONSTRAINT FK_ON_ITEM_ID_02 FOREIGN KEY (ITEM_ID)
 REFERENCES PROD_DUMMY(ITEM_ID),
 CONSTRAINT FK_ON_SHIP_TO_ID_02 FOREIGN KEY (SHIP_TO_ID)
 REFERENCES CUSTOM_DUMMY(SHIP_TO_ID),
 CONSTRAINT FK_ON_MONTH_ID_02 FOREIGN KEY (MONTH_ID)
 REFERENCES TIME_DUMMY(MONTH_ID));
```

OLAP カタログ・メタデータ用の SQL スクリプト

例 B-5 から例 B-10 に、GLOBALX スター・スキーマの OLAP カタログ・メタデータを定義するスクリプトを示します。このメタデータを使用すると、アナリティック・ワークスペース作成ウィザードでデータベース・スタンダード・フォームのアナリティック・ワークスペースを定義できます。

例 B-5 CHANNEL メタデータの作成用スクリプト

```
EXECUTE CWM2_OLAP_DIMENSION.DROP_DIMENSION('GLOBALX','CHANNEL');
EXECUTE CWM2_OLAP_DIMENSION.CREATE_DIMENSION('GLOBALX','CHANNEL','Channel','CHANNEL',
'Channel','Channel',NULL);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','CHANNEL','Long
Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','CHANNEL','Short
Description','Short Description','Short Description','Short Description',1);
EXECUTE cwm2_olap_hierarchy.create_hierarchy('GLOBALX','CHANNEL','CHANNEL_ROLLUP','Channel
Rollup','Channel Rollup','Channel Rollup','UNSOLVED LEVEL-BASED');
EXECUTE cwm2_olap_dimension.set_default_display_hierarchy('GLOBALX','CHANNEL', CHANNEL_ROLLUP);
```

```

EXECUTE cwm2_olap_level.create_level('GLOBALX','CHANNEL','ALL_CHANNELS','All Channels','All
Channels','All Channels','All Channels');
EXECUTE cwm2_olap_level.create_level('GLOBALX','CHANNEL','CHANNEL','Channel','Channel',
'Channel','Channel');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CHANNEL','CHANNEL_ROLLUP',
'ALL_CHANNELS',NULL);
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CHANNEL','CHANNEL_ROLLUP','CHANNEL',
'ALL_CHANNELS');
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CHANNEL','Long
Description','ALL_CHANNELS','Long Description','Long Description','Long Description','Long
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CHANNEL','Long
Description','CHANNEL','Long Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CHANNEL','Short
Description','ALL_CHANNELS','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CHANNEL','Short
Description','CHANNEL','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CHANNEL','CHANNEL_ROLLUP',
'ALL_CHANNELS','GLOBALX','CHANNEL_DUMMY','ALL_CHANNELS_ID',NULL);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CHANNEL','CHANNEL_ROLLUP',
'CHANNEL','GLOBALX','CHANNEL_DUMMY','CHANNEL_ID','ALL_CHANNELS_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CHANNEL','Long
Description','CHANNEL_ROLLUP','CHANNEL','Long Description','GLOBALX','CHANNEL_DUMMY','CHANNEL_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CHANNEL','Long Description',
'CHANNEL_ROLLUP','ALL_CHANNELS','Long Description','GLOBALX','CHANNEL_DUMMY','ALL_CHANNELS_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CHANNEL','Short
Description','CHANNEL_ROLLUP','CHANNEL','Short Description','GLOBALX','CHANNEL_DUMMY','CHANNEL_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CHANNEL','Short
Description','CHANNEL_ROLLUP','ALL_CHANNELS','Short Description',
'GLOBALX','CHANNEL_DUMMY','ALL_CHANNELS_DSC');
EXECUTE cwm2_olap_validate.validate_dimension('GLOBALX','CHANNEL');

```

例 B-6 CUSTOMER メタデータの作成用スクリプト

```

EXECUTE CWM2_OLAP_DIMENSION.DROP_DIMENSION('GLOBALX','CUSTOMER');
EXECUTE CWM2_OLAP_DIMENSION.CREATE_DIMENSION('GLOBALX','CUSTOMER','Customer','CUSTOMER',
'Customer','Customer',NULL);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','CUSTOMER','Long
Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','CUSTOMER','Short
Description','Short Description','Short Description','Short Description',1);
EXECUTE cwm2_olap_hierarchy.create_hierarchy('GLOBALX','CUSTOMER','MARKET_ROLLUP','Market
Rollup','Market Rollup','Market Rollup','UNSOLVED LEVEL-BASED');
EXECUTE cwm2_olap_hierarchy.create_hierarchy('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP','Shipments
Rollup','Shipments Rollup','Shipments Rollup','UNSOLVED LEVEL-BASED');

```

```

EXECUTE cwm2_olap_dimension.set_default_display_hierarchy('GLOBALX', 'CUSTOMER','SHIPMENTS_ROLLUP');
EXECUTE cwm2_olap_level.create_level('GLOBALX','CUSTOMER','ALL_CUSTOMERS','All Customers','All
Customers','All Customers','All Customers');
EXECUTE cwm2_olap_level.create_level('GLOBALX','CUSTOMER','REGION','Region',
'Region','Region','Region');
EXECUTE cwm2_olap_level.create_level('GLOBALX','CUSTOMER','WAREHOUSE',
'Warehouse','Warehouse','Warehouse','Warehouse');
EXECUTE cwm2_olap_level.create_level('GLOBALX','CUSTOMER','TOTAL_MARKET','Total Market','Total
Market','Total Market','Total Market');
EXECUTE cwm2_olap_level.create_level('GLOBALX','CUSTOMER','MARKET_SEGMENT','Market Segment','Market
Segment','Market Segment','Market Segment');
EXECUTE cwm2_olap_level.create_level('GLOBALX','CUSTOMER','ACCOUNT',
'Account','Account','Account','Account');
EXECUTE cwm2_olap_level.create_level('GLOBALX','CUSTOMER','SHIP_TO','Ship To','Ship To','Ship
To','Ship To');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CUSTOMER',
'MARKET_ROLLUP','TOTAL_MARKET',NULL);
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CUSTOMER',
'MARKET_ROLLUP','MARKET_SEGMENT','TOTAL_MARKET');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CUSTOMER','MARKET_ROLLUP',
'ACCOUNT','MARKET_SEGMENT');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CUSTOMER','MARKET_ROLLUP',
'SHIP_TO','ACCOUNT');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP',
'ALL_CUSTOMERS',NULL);
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP',
'REGION','ALL_CUSTOMERS');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP',
'WAREHOUSE','REGION');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP',
'SHIP_TO','WAREHOUSE');
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Long
Description','ALL_CUSTOMERS','Long Description','Long Description','Long Description','Long
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Long
Description','REGION','Long Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Long
Description','WAREHOUSE','Long Description','Long Description','Long Description','Long
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Long
Description','TOTAL_MARKET','Long Description','Long Description','Long Description','Long
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Long
Description','MARKET_SEGMENT','Long Description','Long Description','Long Description','Long
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Long
Description','ACCOUNT','Long Description','Long Description','Long Description','Long Description',1);

```

```

EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Long
Description','SHIP_TO','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Short
Description','ALL_CUSTOMERS','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Short
Description','REGION','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Short
Description','WAREHOUSE','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Short
Description','TOTAL_MARKET','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Short
Description','MARKET_SEGMENT','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Short
Description','ACCOUNT','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','CUSTOMER','Short
Description','SHIP_TO','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CUSTOMER','MARKET_ROLLUP',
'TOTAL_MARKET','GLOBALX','CUSTOM_DUMMY','ALL_CUSTOMERS_ID',NULL);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CUSTOMER','MARKET_ROLLUP',
'MARKET_SEGMENT','GLOBALX','CUSTOM_DUMMY','MARKET_SEGMENT_ID','ALL_CUSTOMERS_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CUSTOMER','MARKET_ROLLUP',
'ACCOUNT','GLOBALX','CUSTOM_DUMMY','ACCOUNT_ID','MARKET_SEGMENT_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CUSTOMER','MARKET_ROLLUP',
'SHIP_TO','GLOBALX','CUSTOM_DUMMY','SHIP_TO_ID','ACCOUNT_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP',
'ALL_CUSTOMERS','GLOBALX','CUSTOM_DUMMY','ALL_CUSTOMERS_ID',NULL);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP',
'REGION','GLOBALX','CUSTOM_DUMMY','REGION_ID','ALL_CUSTOMERS_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP',
'WAREHOUSE','GLOBALX','CUSTOM_DUMMY','WAREHOUSE_ID','REGION_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','CUSTOMER','SHIPMENTS_ROLLUP',
'SHIP_TO','GLOBALX','CUSTOM_DUMMY','SHIP_TO_ID','WAREHOUSE_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Long
Description','MARKET_ROLLUP','SHIP_TO','Long Description','GLOBALX','CUSTOM_DUMMY','SHIP_TO_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Long
Description','MARKET_ROLLUP','ACCOUNT','Long Description','GLOBALX','CUSTOM_DUMMY','ACCOUNT_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Long
Description','MARKET_ROLLUP','MARKET_SEGMENT','Long Description','GLOBALX',
'CUSTOM_DUMMY','MARKET_SEGMENT_DSC');

```

```
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Long
Description','MARKET_ROLLUP','TOTAL_MARKET','Long Description','GLOBALX','
CUSTOM_DUMMY','TOTAL_MARKET_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Long
Description','SHIPMENTS_ROLLUP','SHIP_TO','Long Description','GLOBALX','CUSTOM_DUMMY','SHIP_TO_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Long
Description','SHIPMENTS_ROLLUP','WAREHOUSE','Long Description','GLOBALX',
'CUSTOM_DUMMY','WAREHOUSE_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Long
Description','SHIPMENTS_ROLLUP','REGION','Long Description','GLOBALX','CUSTOM_DUMMY','REGION_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Long
Description','SHIPMENTS_ROLLUP','ALL_CUSTOMERS','Long Description','GLOBALX',
'CUSTOM_DUMMY','ALL_CUSTOMERS_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Short
Description','MARKET_ROLLUP','SHIP_TO','Short Description','GLOBALX','CUSTOM_DUMMY','SHIP_TO_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Short
Description','MARKET_ROLLUP','ACCOUNT','Short Description','GLOBALX','CUSTOM_DUMMY','ACCOUNT_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Short
Description','MARKET_ROLLUP','MARKET_SEGMENT','Short Description','GLOBALX',
'CUSTOM_DUMMY','MARKET_SEGMENT_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Short
Description','MARKET_ROLLUP','TOTAL_MARKET','Short Description','GLOBALX',
'CUSTOM_DUMMY','TOTAL_MARKET_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Short
Description','SHIPMENTS_ROLLUP','SHIP_TO','Short Description','GLOBALX',
'CUSTOM_DUMMY','SHIP_TO_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Short
Description','SHIPMENTS_ROLLUP','WAREHOUSE','Short Description','GLOBALX',
'CUSTOM_DUMMY','WAREHOUSE_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Short
Description','SHIPMENTS_ROLLUP','REGION','Short Description','GLOBALX','CUSTOM_DUMMY','REGION_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','CUSTOMER','Short
Description','SHIPMENTS_ROLLUP','ALL_CUSTOMERS','Short Description','GLOBALX',
'CUSTOM_DUMMY','ALL_CUSTOMERS_DSC');
EXECUTE cwm2_olap_validate.validate_dimension('GLOBALX','CUSTOMER');
```

例 B-7 PRODUCT メタデータの作成用スクリプト

```
EXECUTE CWM2_OLAP_DIMENSION.DROP_DIMENSION('GLOBALX','PRODUCT');
EXECUTE CWM2_OLAP_DIMENSION.CREATE_DIMENSION('GLOBALX','PRODUCT','Product','PRODUCT',
'Product','Product',NULL);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','PRODUCT','Long
Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','PRODUCT','Short
Description','Short Description','Short Description','Short Description',1);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','PRODUCT','Package',
'Package','Package','Package',0);
```

```

EXECUTE cwm2_olap_hierarchy.create_hierarchy('GLOBALX','PRODUCT','PRODUCT_ROLLUP','Product
Rollup','Product Rollup','Product Rollup','UNSOLVED LEVEL-BASED');
EXECUTE cwm2_olap_dimension.set_default_display_hierarchy('GLOBALX','PRODUCT','PRODUCT_ROLLUP');
EXECUTE cwm2_olap_level.create_level('GLOBALX','PRODUCT','TOTAL_PRODUCT','Total Product','Total
Product','Total Product','Total Product');
EXECUTE cwm2_olap_level.create_level('GLOBALX','PRODUCT','CLASS','Class','Class','Class','Class');
EXECUTE cwm2_olap_level.create_level('GLOBALX','PRODUCT','FAMILY',
'Family','Family','Family','Family');
EXECUTE cwm2_olap_level.create_level('GLOBALX','PRODUCT','ITEM','Item','Item','Item','Item');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','PRODUCT','PRODUCT_ROLLUP',
'TOTAL_PRODUCT',NULL);
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','PRODUCT','PRODUCT_ROLLUP','
CLASS','TOTAL_PRODUCT');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','PRODUCT','PRODUCT_ROLLUP',
'FAMILY','CLASS');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','PRODUCT','PRODUCT_ROLLUP','ITEM','FAMILY');
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT','Long
Description','TOTAL_PRODUCT','Long Description','Long Description','Long Description','Long
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT','Long
Description','CLASS','Long Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT','Long
Description','FAMILY','Long Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT','Long
Description','ITEM','Long Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT','Short
Description','TOTAL_PRODUCT','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT','Short
Description','CLASS','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT','Short
Description','FAMILY','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT','Short
Description','ITEM','Short Description','Short Description','Short Description','Short
Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','PRODUCT',
'Package','ITEM','Package','Package','Package','Package',0);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','PRODUCT','PRODUCT_ROLLUP',
'TOTAL_PRODUCT','GLOBALX','PROD_DUMMY','TOTAL_PRODUCT_ID',NULL);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','PRODUCT',
'PRODUCT_ROLLUP','CLASS','GLOBALX','PROD_DUMMY','CLASS_ID','TOTAL_PRODUCT_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','PRODUCT',
'PRODUCT_ROLLUP','FAMILY','GLOBALX','PROD_DUMMY','FAMILY_ID','CLASS_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','PRODUCT','PRODUCT_ROLLUP',
'ITEM','GLOBALX','PROD_DUMMY','ITEM_ID','FAMILY_ID');

```

```
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT','Long
Description','PRODUCT_ROLLUP','ITEM','Long Description','GLOBALX','PROD_DUMMY','ITEM_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT','Long
Description','PRODUCT_ROLLUP','FAMILY','Long Description','GLOBALX','PROD_DUMMY','FAMILY_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT','Long
Description','PRODUCT_ROLLUP','CLASS','Long Description','GLOBALX','PROD_DUMMY','CLASS_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT','Long
Description','PRODUCT_ROLLUP','TOTAL_PRODUCT','Long Description','GLOBALX',
'PROD_DUMMY','TOTAL_PRODUCT_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT','Short
Description','PRODUCT_ROLLUP','ITEM','Short Description','GLOBALX','PROD_DUMMY','ITEM_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT','Short
Description','PRODUCT_ROLLUP','FAMILY','Short Description','GLOBALX','PROD_DUMMY','FAMILY_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT','Short
Description','PRODUCT_ROLLUP','CLASS','Short Description','GLOBALX','PROD_DUMMY','CLASS_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT','Short
Description','PRODUCT_ROLLUP','TOTAL_PRODUCT','Short Description','GLOBALX','PROD_DUMMY',
'TOTAL_PRODUCT_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','PRODUCT',
'Package','PRODUCT_ROLLUP','ITEM','Package','GLOBALX','PROD_DUMMY','ITEM_PACKAGE_ID');
EXECUTE cwm2_olap_validate.validate_dimension('GLOBALX','PRODUCT');
```

例 B-8 TIME メタデータの作成用スクリプト

```
EXECUTE CWM2_OLAP_DIMENSION.DROP_DIMENSION('GLOBALX','TIME');
EXECUTE CWM2_OLAP_DIMENSION.CREATE_DIMENSION('GLOBALX','TIME','Time','TIME','Time','Time','Time');
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','TIME','Long
Description','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','TIME','Short
Description','Short Description','Short Description','Short Description',1);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','TIME','End Date','End
Date','End Date','End Date',1);
EXECUTE cwm2_olap_dimension_attribute.create_dimension_attribute_2('GLOBALX','TIME','Time
Span','Timespan','Timespan','Timespan',1);
EXECUTE cwm2_olap_hierarchy.create_hierarchy('GLOBALX','TIME','CALENDAR',
'Calendar','Calendar','Calendar','UNSOLVED LEVEL-BASED');
EXECUTE cwm2_olap_dimension.set_default_display_hierarchy('GLOBALX','TIME','CALENDAR');
EXECUTE cwm2_olap_level.create_level('GLOBALX','TIME','YEAR','Year','Year','Year','Year');
EXECUTE cwm2_olap_level.create_level('GLOBALX','TIME','QUARTER',
'Quarter','Quarter','Quarter','Quarter');
EXECUTE cwm2_olap_level.create_level('GLOBALX','TIME','MONTH','Month','Month','Month','Month');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','TIME','CALENDAR','YEAR',NULL);
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','TIME','CALENDAR','QUARTER','YEAR');
EXECUTE cwm2_olap_level.add_level_to_hierarchy('GLOBALX','TIME','CALENDAR','MONTH','QUARTER');
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Long
Description','YEAR','Long Description','Long Description','Long Description','Long Description',1);
```



```

EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Long
Description','QUARTER','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Long
Description','MONTH','Long Description','Long Description','Long Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Short
Description','YEAR','Short Description','Short Description','Short Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Short
Description','QUARTER','Short Description','Short Description','Short Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Short
Description','MONTH','Short Description','Short Description','Short Description',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','End Date','YEAR','End
Date','End Date','End Date',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','End Date','QUARTER','End
Date','End Date','End Date',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','End Date','MONTH','End
Date','End Date','End Date',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Time Span','YEAR','Time
Span','Timespan','Timespan',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Time
Span','QUARTER','Time Span','Timespan','Timespan',1);
EXECUTE cwm2_olap_level_attribute.create_level_attribute_2('GLOBALX','TIME','Time Span','MONTH','Time
Span','Timespan','Timespan',1);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','TIME','CALENDAR',
'YEAR','GLOBALX','TIME_DUMMY','YEAR_ID',NULL);
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','TIME','CALENDAR',
'QUARTER','GLOBALX','TIME_DUMMY','QUARTER_ID','YEAR_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVEL('GLOBALX','TIME','CALENDAR',
'MONTH','GLOBALX','TIME_DUMMY','MONTH_ID','QUARTER_ID');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Long
Description','CALENDAR','MONTH','Long Description','GLOBALX','TIME_DUMMY','MONTH_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Long Description',
'CALENDAR','QUARTER','Long Description','GLOBALX','TIME_DUMMY','QUARTER_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Long Description',
'CALENDAR','YEAR','Long Description','GLOBALX','TIME_DUMMY','YEAR_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Short Description',
'CALENDAR','MONTH','Short Description','GLOBALX','TIME_DUMMY','MONTH_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Short Description',
'CALENDAR','QUARTER','Short Description','GLOBALX','TIME_DUMMY','QUARTER_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Short Description',
'CALENDAR','YEAR','Short Description','GLOBALX','TIME_DUMMY','YEAR_DSC');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','End Date', 'CALENDAR',
'MONTH','End Date','GLOBALX','TIME_DUMMY','MONTH_END_DATE');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','End Date',
'CALENDAR','QUARTER','End Date','GLOBALX','TIME_DUMMY','QUARTER_END_DATE');

```

```
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','End Date',
'CALENDAR','YEAR','End Date','GLOBALX','TIME_DUMMY','YEAR_END_DATE');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Time Span',
'CALENDAR','MONTH','Time Span','GLOBALX','TIME_DUMMY','MONTH_TIMESPAN');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Time Span',
'CALENDAR','QUARTER','Time Span','GLOBALX','TIME_DUMMY','QUARTER_TIMESPAN');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_DIMTBL_HIERLEVELATTR('GLOBALX','TIME','Time Span',
'CALENDAR','YEAR','Time Span','GLOBALX','TIME_DUMMY','YEAR_TIMESPAN');
EXECUTE cwm2_olap_validate.validate_dimension('GLOBALX','TIME');
```

例 B-9 UNITS キューブの作成用スクリプト

```
EXECUTE CWM2_OLAP_CUBE.DROP_CUBE('GLOBALX','UNITS_CUBE');
EXECUTE CWM2_OLAP_CUBE.CREATE_CUBE('GLOBALX','UNITS_CUBE','UNITS_CUBE','UNITS_CUBE','UNITS_CUBE');
EXECUTE CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBALX','UNITS_CUBE','GLOBALX','CHANNEL');
EXECUTE CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBALX','UNITS_CUBE','GLOBALX','CUSTOMER');
EXECUTE CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBALX','UNITS_CUBE','GLOBALX','PRODUCT');
EXECUTE CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBALX','UNITS_CUBE','GLOBALX','TIME');
EXECUTE CWM2_OLAP_MEASURE.CREATE_MEASURE('GLOBALX','UNITS_CUBE','UNITS','UNITS','Units Sold','Units
Sold');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_LEVELKEY('GLOBALX','UNITS_CUBE','GLOBALX','UNITS_DUMMY',
'LOWESTLEVEL','DIM:GLOBALX.CHANNEL/HIER:CHANNEL_ROLLUP/LVL:CHANNEL/COL:CHANNEL_ID;
DIM:GLOBALX.CUSTOMER/HIER:MARKET_ROLLUP/LVL:SHIP_TO/COL:SHIP_TO_ID;
DIM:GLOBALX.PRODUCT/HIER:PRODUCT_ROLLUP/LVL:ITEM/COL:ITEM_ID;
DIM:GLOBALX.TIME/HIER:CALENDAR/LVL:MONTH/ COL:MONTH_ID;');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_LEVELKEY('GLOBALX','UNITS_CUBE','GLOBALX',
'UNITS_DUMMY','LOWESTLEVEL','DIM:GLOBALX.CHANNEL/HIER:CHANNEL_ROLLUP/LVL:CHANNEL/COL:CHANNEL_ID;
DIM:GLOBALX.CUSTOMER/HIER:SHIPMENTS_ROLLUP/LVL:SHIP_TO/COL:SHIP_TO_ID;
DIM:GLOBALX.PRODUCT/HIER:PRODUCT_ROLLUP/LVL:ITEM/COL:ITEM_ID;
DIM:GLOBALX.TIME/HIER:CALENDAR/LVL:MONTH/COL:MONTH_ID;');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_MEASURE('GLOBALX','UNITS_CUBE','UNITS','GLOBALX',
'UNITS_DUMMY','UNITS','DIM:GLOBALX.CHANNEL/HIER:CHANNEL_ROLLUP/LVL:CHANNEL/COL:CHANNEL_ID;
DIM:GLOBALX.CUSTOMER/HIER:MARKET_ROLLUP/LVL:SHIP_TO/COL:SHIP_TO_ID;
DIM:GLOBALX.PRODUCT/HIER:PRODUCT_ROLLUP/LVL:ITEM/COL:ITEM_ID;
DIM:GLOBALX.TIME/HIER:CALENDAR/LVL:MONTH/COL:MONTH_ID;');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_MEASURE('GLOBALX','UNITS_CUBE','UNITS','GLOBALX',
'UNITS_DUMMY','UNITS','DIM:GLOBALX.CHANNEL/HIER:CHANNEL_ROLLUP/LVL:CHANNEL/COL:CHANNEL_ID;
DIM:GLOBALX.CUSTOMER/HIER:SHIPMENTS_ROLLUP/LVL:SHIP_TO/COL:SHIP_TO_ID;
DIM:GLOBALX.PRODUCT/HIER:PRODUCT_ROLLUP/LVL:ITEM/COL:ITEM_ID;
DIM:GLOBALX.TIME/HIER:CALENDAR/LVL:MONTH/COL:MONTH_ID;');
EXECUTE cwm2_olap_validate.validate_Cube('GLOBALX','UNITS_CUBE');
```

例 B-10 PRICE キューブの作成用スクリプト

```
EXECUTE CWM2_OLAP_CUBE.DROP_CUBE('GLOBALX','PRICE_CUBE');
EXECUTE CWM2_OLAP_CUBE.CREATE_CUBE('GLOBALX','PRICE_CUBE','PRICE_CUBE','PRICE_CUBE','PRICE_CUBE');
```

```
EXECUTE CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBALX','PRICE_CUBE','GLOBALX','TIME');EXECUTE
CWM2_OLAP_CUBE.ADD_DIMENSION_TO_CUBE('GLOBALX','PRICE_CUBE','GLOBALX','PRODUCT');
EXECUTE CWM2_OLAP_MEASURE.CREATE_MEASURE('GLOBALX','PRICE_CUBE','UNIT_COST','UNIT COST','Unit
Cost','Unit Cost');
EXECUTE CWM2_OLAP_MEASURE.CREATE_MEASURE('GLOBALX','PRICE_CUBE','UNIT_PRICE','UNIT PRICE','Unit
Price','Unit Price');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_LEVELKEY('GLOBALX','PRICE_CUBE','GLOBALX',
'PRICE_AND_COST_DUMMY','LOWESTLEVEL', 'DIM:GLOBALX.PRODUCT/HIER:PRODUCT_ROLLUP/LVL:ITEM/COL:ITEM_ID;
DIM:GLOBALX.TIME/HIER:CALENDAR/LVL:MONTH/COL:MONTH_ID;');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_MEASURE('GLOBALX','PRICE_CUBE','UNIT_COST','GLOBALX',
'PRICE_AND_COST_DUMMY','UNIT_COST', 'DIM:GLOBALX.PRODUCT/HIER:PRODUCT_ROLLUP/LVL:ITEM/COL:ITEM_ID;
DIM:GLOBALX.TIME/HIER:CALENDAR/LVL:MONTH/COL:MONTH_ID;');
EXECUTE CWM2_OLAP_TABLE_MAP.MAP_FACTTBL_MEASURE('GLOBALX','PRICE_CUBE','UNIT_PRICE','GLOBALX',
'PRICE_AND_COST_DUMMY','UNIT_PRICE', 'DIM:GLOBALX.PRODUCT/HIER:PRODUCT_ROLLUP/LVL:ITEM/COL:ITEM_ID;
DIM:GLOBALX.TIME/HIER:CALENDAR/LVL:MONTH/COL:MONTH_ID;');
EXECUTE cwm2_olap_validate.validate_Cube('GLOBALX', 'PRICE_CUBE');
```

アナリティック・ワークスペースのデータベース・スタンダード・フォーム

データベース・スタンダード・フォームは、様々な Oracle OLAP ユーティリティで管理できる、アナリティック・ワークスペースのオブジェクトを記述する一連の規則です。この付録では、データベース・スタンダード・フォーム規則について説明します。この付録では、次の項目について説明します。

- データベース・スタンダード・フォームの概要
- オブジェクトのネーミング規則
- ワークスペース・オブジェクトのプロパティ
- 実装クラスオブジェクト
- カタログ・クラスオブジェクト
- 機能クラスオブジェクト
- 拡張クラスオブジェクト

データベース・スタンダード・フォームの概要

データベース・スタンダード・フォームに準拠するアナリティック・ワークスペースには、キューブ、ディメンションおよびメジャーの論理モデルを実装するオブジェクトがあります。スタンダード・フォームには、ワークスペース・オブジェクトのネーミング規則が含まれ、論理モデルを、それを実装するワークスペース・オブジェクトに関連付けるオブジェクトのプロパティを指定します。

データベース・スタンダード・フォームの目的

スタンダード・フォーム規則の目的は、スタンダード・フォームに基づいた論理モデルおよびワークスペースの実装を、関連する Oracle OLAP ユーティリティでできるようにすることです。これらのユーティリティは、スタンダード・フォームに基づいたデータおよび

メタデータを処理するので、各ユーティリティは相互に互換性があります。したがって、これらのユーティリティを使用する DBA は、複数の分析ツールを介してアクセスできるアナリティック・ワークスペースを作成およびメンテナンスできます。現時点では、OLAP API、BI Beans および Discoverer を介してアクセスできます。

Analytic Workspace Manager では、次のユーティリティを備えています。ユーティリティには、スタンダード・フォームのワークスペースを作成するものと、スタンダード・フォーム・ワークスペースが作成済の段階で使用するものがあります。

- アナリティック・ワークスペース作成ウィザードでは、OLAP カタログ・メタデータからスタンダード・フォームの新しいワークスペースを作成します。
- アナリティック・ワークスペースのリフレッシュ・ウィザードでは、OLAP カタログ・メタデータからスタンダード・フォームの既存のワークスペースをリフレッシュします。
- OLAP API および BI Beans の有効化ウィザードでは、OLAP API および BI Beans によるスタンダード・フォーム・ワークスペースへのアクセスを有効化します。
- Discoverer の有効化ウィザードでは、Discoverer によるスタンダード・フォーム・ワークスペースへのアクセスを有効化します。

Analytic Workspace Manager では、PL/SQL パッケージの DBMS_AWM を使用して、スタンダード・フォームのアナリティック・ワークスペースを作成およびリフレッシュし、OLAP API によるアクセスを有効化します。Analytic Workspace Manager を使用すると、アナリティック・ワークスペースを管理できます。また、DBMS_AWM プロシージャを使用して、独自のスクリプトを作成できます。

データベース・スタンダード・フォームの対象読者

通常は、付属のツールおよびプロシージャを使用してスタンダード・フォームのワークスペースを作成、管理および有効化するため、スタンダード・フォームの詳細な知識が必要になることはありません。ただし、次のような状況では詳細な知識が必要になります。

- 既存のスタンダード・フォームのワークスペースに手動で追加を行う場合。たとえば、OLAP カタログ・メタデータにディメンションの複数説明が含まれておらず、DBA が OLAP カタログではなくワークスペースに複数説明を追加する場合などです。
- スタンダード・フォームのワークスペースを使用するアプリケーションを開発する際に、どのメジャーを特定のワークスペースにおける分析に使用できるか、そのメジャーをどのようにディメンション化するか、どのレベルおよび階層が定義されているか、などの情報をそのアプリケーションの実行時に取得させたい場合。

これらの要件が発生した場合にスタンダード・フォーム・ワークスペースの規則を理解できるよう、この付録では、スタンダード・フォームについて説明します。これらの規則を理解するには、多次元な OLAP の概念および OLAP DML の使用に習熟している必要があります。

論理モデルとワークスペース・オブジェクト

スタンダード・フォームの論理モデルには、キューブ、メジャー、ディメンションの他に、ディメンションに関連付けられている階層、レベルおよび属性が含まれます。キューブは、キューブに含まれるメジャーの親と見なされ、ディメンションは、その階層、レベルおよび属性の親と見なされます。キューブには次元性があります。つまり、キューブはディメンションのリストに関連付けられています。

キューブの実装

論理キューブを実装するプライマリ・ワークスペース・オブジェクトは、*cubedef* ディメンションと呼ばれるワークスペース・ディメンションです。このディメンションの値は、キューブのディメンションの名前です。

論理キューブを実装するセカンダリ・ワークスペース・オブジェクトは、*aggmap* (*comspec aggmap* と呼ばれる) およびコンポジット (*loopspec* コンポジットと呼ばれる) です。

これらのオブジェクトの詳細は、C-15 ページの「[キューブのオブジェクト](#)」を参照してください。

メジャーの実装

論理メジャーを実装するプライマリ・ワークスペース・オブジェクトは、*measuredef* オブジェクトと呼ばれるワークスペースの変数、式またはリレーションです。このオブジェクトの値は、論理メジャーの値です。

メジャーのセカンダリ・ワークスペース・オブジェクトは、キューブの *comspec aggmap* のみです。

これらのオブジェクトの詳細は、C-17 ページの「[メジャーのオブジェクト](#)」を参照してください。

ディメンションの実装

論理ディメンションを実装するプライマリ・ワークスペース・オブジェクトは、*dimdef* ディメンションと呼ばれるワークスペース・ディメンションです。このディメンションの値は、論理ディメンションの値です。

ディメンションの階層およびレベルには、独自のプライマリ・オブジェクトはありません。かわりに、次のオブジェクトによって実装が提供されます。

- *hierlist* ディメンションは、ディメンションの階層を示します。
- *levellist* ディメンションは、ディメンションのレベルを示します。
- *parentrel* リレーションは、ディメンションの各メンバーの親を記録します。
- *levelrel* リレーションは、ディメンションの各メンバーのレベルを記録します。
- *hier_levels* 値セットは、各階層のレベルを記録します。

ディメンションの属性を実装するプライマリ・ワークスペース・オブジェクトは、*attrdef* オブジェクトと呼ばれるワークスペースの変数、式またはリレーションです。このオブジェクトの値は、論理属性の値です。

これらのオブジェクトの詳細は、C-18 ページの「[ディメンションのオブジェクト](#)」を参照してください。

ワークスペース・オブジェクトのクラス

スタンダード・フォームの各ワークスペース・オブジェクトは、次の 4 つのクラスのいずれかに属します。

- **実装クラス。** このクラスのオブジェクトは、論理モデルを実装します。このオブジェクトには、C-3 ページの「[論理モデルとワークスペース・オブジェクト](#)」で説明するすべてのワークスペース・オブジェクトが含まれます。たとえば、*cubedef*、*measuredef*、*dimdef* および *hierlist* などです。
- **カタログ・クラス。** このクラスのオブジェクトは、論理モデルに関する情報を保持します。このオブジェクトには、ワークスペース内のすべてのキューブのリスト、ワークスペース内のすべてのメジャーのリスト、ワークスペース内のすべてのディメンションのリスト、および様々なユーティリティの作業を容易にするその他のリストが含まれます。
- **機能クラス。** このクラスのオブジェクトは、論理モデル内の特定のオブジェクトに関する情報を保持します。たとえば、すべての論理オブジェクトの説明を格納するオブジェクトや、オブジェクトをユーザーに表示するかどうかを示すオブジェクトなどです。
- **拡張クラス。** このクラスのオブジェクトは、Oracle OLAP ユーティリティで定義およびメンテナンスされます。このオブジェクトは、スタンダード・フォームへの独自の拡張であり、リリース間でのメンテナンスについて Oracle では保証しません。

拡張クラスのオブジェクトは、定義、変更または依存しないください。

ワークスペース・オブジェクトのプロパティ

スタンダード・フォームの重要な特徴は、論理モデルの実装でワークスペース・オブジェクトの OLAP DML プロパティに依存する点です。OLAP DML プロパティは、OLAP DML の PROPERTY コマンドを使用して割り当てます。

オブジェクトのネーミング規則

OLAP DML で定められた規則を除き、スタンダード・フォームの論理モデルを実装するワークスペース・オブジェクトの名前に関する制限事項はありません。ただし、論理オブジェクトの場合は、スタンダード・フォームで厳密なネーミング規則が定められています。これは、スタンダード・フォームに依存するユーティリティが論理名でオブジェクトを参照するためです。

論理名に関するスタンダード・フォームのネーミング規則は、Oracle Database のネーミング規則に準拠します。ネーミング規則では、論理名が重複しないネームスペースを確立し、フルネームを作成してネームスペースの構成を反映するための規則を提供します。論理名は、フルネームと区別するために「単純論理名」と呼ばれる場合があります。

論理名

一般的に、キューブやディメンションなどのオブジェクトの単純論理名は、わずかな違いはあるものの SQL の単純式の規則に準拠しています。スタンダード・フォームの論理名の規則では、名前は次の要件を満たす必要があります。

1. バイト数は 1 ～ 30 バイト。
2. Oracle の予約語は使用できない。
3. 大 / 小文字を区別しない。
4. 引用符を含めることはできない。
5. データベース・キャラクタ・セットのアルファベット文字で開始する必要がある。
6. データベース・キャラクタ・セットの英数字、アンダースコア (_)、ドル記号 (\$) および番号記号 (#) 以外は使用できない。ただし、ドル記号または番号記号は使用しないことをお勧めします。データベース・キャラクタ・セットにマルチバイト文字が含まれている場合、各論理名にシングルバイト文字を最低 1 つ含めることをお勧めします。

ワークスペース・オブジェクトの `AW$LOGICAL_NAME` プロパティには、それを実装するオブジェクトの単純論理名が含まれます。たとえば、`PRODUCT` などの単純論理名があります。

ネームスペースの構成

スタンダード・フォームのネーミング規則では、次のネームスペースを定義する論理オブジェクトの構成を規定しています。

- **スキーマ**。キューブおよびディメンションの論理名は、アナリティック・ワークスペースを所有するスキーマ内で一意である必要があります。
- **キューブ**。メジャーの論理名は、特定のキューブ内で一意である必要があります。
- **ディメンション**。階層の論理名は、ディメンション内で一意である必要があります。レベルの論理名は、ディメンション内で一意である必要があります。属性の論理名は、ディメンション内で一意である必要があります。特定のディメンション内で、階層はレベルまたは属性と同じ名前を持つことができます。

ネームスペースの構成は、論理オブジェクト間の所有権、親または関係を反映しています。たとえば、メジャーはキューブを親オブジェクトとして持ち、属性はディメンションを親オブジェクトとして持ちます。ワークスペース・オブジェクトの `AW$PARENT_NAME` プロパティでは、このような関係を記録します。

単純論理名とフルネーム

単純論理名はネームスペースの外部では一意でないため、スタンダード・フォーム規則では論理オブジェクトごとにフルネームを指定します。フルネームは、単純論理名を含みますが、オブジェクトの所属先のネームスペースおよびオブジェクト型も示します。次に、属性のフルネームの例を示します。属性の単純名は `TIME_SPAN`、属性の親オブジェクトは `TIME` というディメンションです。

```
GLOBAL_AW.TIME.TIME_SPAN.ATTRIBUTE
```

フルネームの最後の構成要素はオブジェクト型です。この例の場合、オブジェクト型は `ATTRIBUTE` です。指定可能な型は、すべて *all_objtypes* ディメンションで示されます (C-28 ページの「[ALL_OBJTYPES ディメンション](#)」を参照)。

フルネームは、様々なオブジェクト型を示すカタログ・クラスのオブジェクトで使用されます。たとえば、*all_dimensions*、*all_cubes* および *all_attributes* ディメンションの値は、論理オブジェクトのフルネームになります。

ワークスペース・オブジェクトのプロパティ

C-4 ページの「[ワークスペース・オブジェクトのプロパティ](#)」では、スタンダード・フォームのプロパティの使用について概説しました。プロパティは、ワークスペースのオブジェクト単位で論理オブジェクトを実装する主な方法です。プロパティは、OLAP DML の `PROPERTY` コマンドを使用してワークスペース・オブジェクトに作成します。

スタンダード・フォームのワークスペース・オブジェクトは、適切に定義されたプロパティを持ちます。プロパティは、次の 3 つのグループに分類されます。

- 実装クラスのオブジェクトに固有のプロパティ。
- すべてのワークスペース・オブジェクトのシステム・プロパティ。

このプロパティは、Oracle OLAP ユーティリティ (DBMS_AWM または *Analytic Workspace Manager* で提供されるユーティリティ) を使用して作成し、値の指定を行います。このプロパティは、変更または削除しないでください。
- すべてのワークスペース・オブジェクトのロール・プロパティ。

スタンダード・フォームのオブジェクトはすべて、`AW$ROLE` というプロパティを持ちます。このプロパティは、スタンダード・フォームのオブジェクトで果たされるロール (または機能) を示します。

実装クラスのオブジェクトに固有のプロパティ

論理名および親名のプロパティは、すべての実装クラスのオブジェクトにあります。オブジェクトのロールに応じて、3 つの追加プロパティを指定する場合があります。

表 C-1 に、実装クラスのプロパティおよび各プロパティの説明を示します。

表 C-1 実装クラスのプロパティ

プロパティ	説明
AW\$LOGICAL_NAME	このワークスペース・オブジェクトで実装される論理オブジェクトの単純論理名。ロールが CUBEDEF、MEASUREDEF、DIMDEF および ATTRDEF のオブジェクトに対してのみ値を設定します。実装クラスのその他のすべてのロールについては、プロパティは存在しますが、値は NA です。
AW\$PARENT_NAME	このワークスペース・オブジェクトで実装される、論理オブジェクトの親の単純論理名。ロールが CUBEDEF および DIMDEF の場合を除き、すべての実装クラスのオブジェクトに対して値を設定します。これら 2 つのロールの場合、オブジェクトに親がないため、値は NA です。
AW\$LOOPSPEC	ロールが CUBEDEF のオブジェクトの場合、キューブのコンポジット名。これは、ワークスペース・オブジェクトの名前であり、オブジェクトの論理名ではありません。その他のすべてのロールの場合、このプロパティはありません。
AW\$COMPSPEC	ロールが MEASUREDEF のオブジェクトの場合、メジャーの AGGMAP オブジェクトの名前。これは、ワークスペース・オブジェクトの名前であり、オブジェクトの論理名ではありません。その他のすべてのロールの場合、このプロパティはありません。
AW\$TYPE	<p>ロールが DIMDEF および ATTRDEF のオブジェクトの場合、ディメンションまたは属性のタイプ。その他のすべてのロールの場合、このプロパティはありません。</p> <p>ロールが DIMDEF の場合、このプロパティは、ディメンションが時間ディメンションかどうかを示します。値は、TIME または NA です。</p> <p>ロールが ATTRDEF の場合、このプロパティは、Oracle OLAP による属性の特別な使用方法を示します。特別な使用方法を示す値は、DEFAULT_ORDER、END_DATE、TIME_SPAN、MEMBER_LONG_DESCRIPTION、MEMBER_SHORT_DESCRIPTION、MEMBER_VISIBLE です。値が USER または NA の場合、属性は Oracle OLAP で特別な意味を持ちません。</p>

すべてのワークスペース・オブジェクトのシステム・プロパティ

スタンダード・フォームの一部であるワークスペース・オブジェクトはすべて、4 つのシステム・プロパティを持ちます。

表 C-2 に、システム・プロパティおよび各プロパティの説明を示します。

表 C-2 システム・プロパティ

プロパティ	説明
AW\$CLASS	ワークスペース・オブジェクトのクラス。指定可能な値は、IMPLEMENTATION、CATALOGS、FEATURES および EXTENSIONS です。これらのクラスについては、C-4 ページの「ワークスペース・オブジェクトのクラス」を参照してください。
AW\$CREATEDBY	ワークスペース・オブジェクトを作成したエンティティ。たとえば、DBMS_AWM で作成された場合、値は AW\$CREATE です。
AW\$LASTMODIFIED	最後にワークスペース・オブジェクトが登録された日付と時刻。
AW\$STATE	スタンダード・フォームに関連するワークスペース・オブジェクトの状態（たとえば、VALID_MEMBER）。

すべてのワークスペース・オブジェクトのロール・プロパティ

スタンダード・フォームの一部であるワークスペース・オブジェクトはすべて、ロール・プロパティを持ちます。

表 C-3 に、ロール・プロパティを示します。

表 C-3 ロール・プロパティ

プロパティ	説明
AW\$ROLE	オブジェクトで実行されるロール（機能）。指定可能な値は、オブジェクト・クラスごとに異なります。プロパティ値の詳細は、C-8 ページの「実装クラスのオブジェクトのロール・プロパティ値」、C-9 ページの「カタログ・クラス・オブジェクトのロール・プロパティ値」、C-12 ページの「機能クラス・オブジェクトのロール・プロパティ値」、C-13 ページの「拡張クラス・オブジェクトのロール・プロパティ値」を参照してください。

実装クラスのオブジェクトのロール・プロパティ値

AW\$ROLE プロパティは、ワークスペース・オブジェクトによって実行される機能（ロール）を示します。実装クラスのオブジェクトの場合、ロールは論理モデルの基本的なビルディング・ブロック（キューブ、メジャー、ディメンションなど）を示します。

スタンダード・フォームのワークスペースで同じロールを持つ実装クラスのオブジェクトは、複数存在できます。論理モデルのディメンションごとに DIMDEF ロールを持つオブジェクトが 1 つずつあるため、このロールを持つオブジェクトが複数存在する場合などです。

表 C-4 に、指定可能な値と各ロールの説明を示します。

表 C-4 ロール・プロパティ値：実装クラス

ロール・プロパティ値	ロールの説明
CUBEDEF	AW\$LOGICAL_NAME プロパティで論理名が指定されているキューブを実装します。このロールを持つオブジェクトについては、C-15 ページの「 Cubedef ディメンション 」を参照してください。
MEASUREDEF	AW\$LOGICAL_NAME プロパティで論理名が指定されているメジャーを実装します。このロールを持つオブジェクトについては、C-17 ページの「 Measuredef オブジェクト 」を参照してください。
DIMDEF	AW\$LOGICAL_NAME プロパティで論理名が指定されているディメンションを実装します。このロールを持つオブジェクトについては、C-18 ページの「 Dimdef ディメンション 」を参照してください。
HIERLIST	AW\$PARENT_NAME プロパティで名前が指定されている、ディメンションの階層の名前を示します。このロールを持つオブジェクトについては、C-19 ページの「 Hierlist ディメンション 」を参照してください。
LEVLLIST	AW\$PARENT_NAME プロパティで名前が指定されている、ディメンションのレベルの名前を示します。このロールを持つオブジェクトについては、C-20 ページの「 Levellist ディメンション 」を参照してください。
MEMBER_LEVELREL	AW\$PARENT_NAME プロパティで名前が指定されている、ディメンションの各メンバーのレベルを記録します。このロールを持つオブジェクトについては、C-21 ページの「 Member_Levelrel リレーション 」を参照してください。
MEMBER_PARENTREL	AW\$PARENT_NAME プロパティで名前が指定されている、ディメンションの各メンバーの親を記録します。このロールを持つオブジェクトについては、C-21 ページの「 Member_Parentrel リレーション 」を参照してください。
HIER_LEVELS	AW\$PARENT_NAME プロパティで名前が指定されている、ディメンションの各階層に含まれているレベルを示します。このロールを持つオブジェクトについては、C-22 ページの「 Hier_Levels 値セット 」を参照してください。
ATTRDEF	AW\$LOGICAL_NAME プロパティで論理名が指定されている属性を実装します。このロールを持つオブジェクトについては、C-23 ページの「 Attrdef オブジェクト 」を参照してください。

カタログ・クラス・オブジェクトのロール・プロパティ値

AW\$ROLE プロパティは、ワークスペース・オブジェクトによって実行される機能（ロール）を示します。カタログ・クラスのオブジェクトの場合、様々なロールを持つオブジェクトに

よって、キューブのリスト、オブジェクト型のリスト、メジャーのリストなど、論理モデルに関する情報が提供されます。

スタンダード・フォームのワークスペースで指定されたロールを持つカタログ・クラスオブジェクトは1つのみです。たとえば、ワークスペース内のすべてのディメンションを示すオブジェクトは1つのみです。

表 C-5 に、指定可能な値と各ロールの説明を示します。

表 C-5 ロール・プロパティ値：カタログ・クラス

ロール・プロパティ値	ロールの説明
ALL_OBJECTS	ワークスペースでスタンダード・フォームに登録されているすべてのオブジェクトのフルネームを示します。このロールを持つオブジェクトについては、C-27 ページの「 ALL_OBJECTS ディメンション 」を参照してください。
ALL_CUBES	ワークスペースでスタンダード・フォームに登録されているすべてのキューブのフルネームを示します。このロールを持つオブジェクトについては、C-24 ページの「 ALL_CUBES ディメンション 」を参照してください。
ALL_MEASURES	ワークスペースでスタンダード・フォームに登録されているすべてのメジャーのフルネームを示します。このロールを持つオブジェクトについては、C-25 ページの「 ALL_MEASURES ディメンション 」を参照してください。
ALL_DIMENSIONS	ワークスペースでスタンダード・フォームに登録されているすべてのディメンションのフルネームを示します。このロールを持つオブジェクトについては、C-25 ページの「 ALL_DIMENSIONS ディメンション 」を参照してください。
ALL_HIERARCHIES	ワークスペースでスタンダード・フォームに登録されているすべての階層のフルネームを示します。このロールを持つオブジェクトについては、C-25 ページの「 ALL_HIERARCHIES ディメンション 」を参照してください。
ALL_LEVELS	ワークスペースでスタンダード・フォームに登録されているすべてのレベルのフルネームを示します。このロールを持つオブジェクトについては、C-26 ページの「 ALL_LEVELS ディメンション 」を参照してください。
ALL_ATTRIBUTES	ワークスペースでスタンダード・フォームに登録されているすべての属性のフルネームを示します。このロールを持つオブジェクトについては、C-27 ページの「 ALL_ATTRIBUTES ディメンション 」を参照してください。

表 C-5 ロール・プロパティ値：カタログ・クラス

ロール・プロパティ値	ロールの説明
ALL_OBJTYPES	現在、スタンダード・フォームでサポートされているオブジェクト型を示します。これには、CUBE、MEASURE、DIMENSION、HIERARCHY、LEVEL および ATTRIBUTE があります。このロールを持つオブジェクトについては、C-28 ページの「 ALL_OBJTYPES ディメンション 」を参照してください。
ALL_DESCTYPES	現在、スタンダード・フォームでサポートされている説明のタイプを示します。これには、SHORT、LONG および PLURAL があります。このロールを持つオブジェクトについては、C-29 ページの「 ALL_DESCTYPES ディメンション 」を参照してください。
ALL_ATTRTYPES	現在、スタンダード・フォームでサポートされている属性のタイプをすべて示します。これは、ATTRDEF ロールを持つオブジェクトの AW\$TYPE プロパティの有効な値です。ALL_ATTRTYPES ロールを持つオブジェクトについては、C-29 ページの「 ALL_ATTRTYPES ディメンション 」を参照してください。
AW_ROLES	現在、スタンダード・フォームでサポートされている、AW\$ROLE プロパティの値をすべて示します。リストには、すべてのクラスのオブジェクトのロールが含まれます。このロールを持つオブジェクトについては、C-29 ページの「 AW_ROLES ディメンション 」を参照してください。
ALL_LANGUAGES	DBA によってワークスペースに含められているすべての言語の名前を示します。このロールを持つオブジェクトについては、C-30 ページの「 ALL_LANGUAGES ディメンション 」を参照してください。
CUBE_MEASURES	ワークスペース内の各キューブに属しているメジャーのフルネームを示します。このロールを持つオブジェクトについては、C-31 ページの「 CUBE_MEASURES 値セット 」を参照してください。
DIM_HIERARCHIES	ワークスペース内の各ディメンションに属している階層のフルネームを示します。このロールを持つオブジェクトについては、C-31 ページの「 DIM_HIERARCHIES 値セット 」を参照してください。
DIM_LEVELS	ワークスペース内の各ディメンションに属しているレベルのフルネームを示します。このロールを持つオブジェクトについては、C-32 ページの「 DIM_LEVELS 値セット 」を参照してください。

表 C-5 ロール・プロパティ値：カタログ・クラス

ロール・プロパティ値	ロールの説明
DIM_ATTRIBUTES	ワークスペース内の各ディメンションに属している属性のフルネームを示します。このロールを持つオブジェクトについては、C-32 ページの「 DIM_ATTRIBUTES 値セット 」を参照してください。
AW_NAMES	各論理キューブ、論理メジャー、論理ディメンションおよび論理属性を実装するワークスペース・オブジェクトの名前を記録します。その他の論理オブジェクトの場合、対応するワークスペース・オブジェクトがないため、値はNAです。このロールを持つオブジェクトについては、C-33 ページの「 AW_NAMES 変数 」を参照してください。
AW_COMPSPECS	ディメンションを参照するすべての AGGMAP オブジェクトの名前をディメンションごとに記録します。このロールを持つオブジェクトについては、C-33 ページの「 AW_COMPSPECS 変数 」を参照してください。
AW_LOOPSPECS	キューブのコンポジットの名前をキューブごとに記録します。このロールを持つオブジェクトについては、C-34 ページの「 AW_LOOPSPECS 変数 」を参照してください。

機能クラス・オブジェクトのロール・プロパティ値

AW\$ROLE プロパティは、ワークスペース・オブジェクトによって実行される機能（ロール）を示します。機能クラスのオブジェクトの場合、説明など、論理オブジェクトの補足的なデータの様々なタイプがロールによって提供されます。

ほとんどのロールでは、スタンダード・フォームのワークスペースに機能クラスのオブジェクトが1つあります。ただし、名前に MEMBER を含むロールの場合、ディメンションごとにオブジェクトが1つあります。

表 C-6 に、指定可能な値および機能クラスのオブジェクトに適用される各ロールの説明を示します。

表 C-6 ロール・プロパティ値：機能クラス

ロール・プロパティ値	ロールの説明
ALL_DESCRIPTIONS	すべてのオブジェクトの簡単な説明、詳細な説明および複数説明を記録します。このロールを持つオブジェクトについては、C-34 ページの「 ALL_DESCRIPTIONS 変数 」を参照してください。
ATTR_INHIER	特定の属性が特定の階層に関連付けられているかどうかを示します。このロールを持つオブジェクトについては、C-35 ページの「 ATTR_INHIER 変数 」を参照してください。

表 C-6 ロール・プロパティ値：機能クラス

ロール・プロパティ値	ロールの説明
DEFAULT_HIER	各ディメンションのデフォルトの階層のフルネームを記録します。このロールを持つオブジェクトについては、C-35 ページの「 DEFAULT_HIER リレーション 」を参照してください。
MEMBER_CREATEDBY	特定のディメンションの各メンバーを作成したエンティティを記録します。このロールを持つオブジェクトについては、C-37 ページの「 Member_Createdby 変数 」を参照してください。
MEMBER_FAMILYREL	特定のディメンションの各階層のファミリー・リレーションを記録します。このロールを持つオブジェクトについては、C-37 ページの「 Member_Familyrel リレーション 」を参照してください。
MEMBER_GID	特定のディメンションの各階層のグルーピング ID を記録します。このロールを持つオブジェクトについては、C-37 ページの「 Member_Gid 変数 」を参照してください。
MEMBER_INHIER	ディメンションの特定のメンバーが特定の階層にあるかどうかを示します。このロールを持つオブジェクトについては、C-36 ページの「 Member_Inhier 変数 」を参照してください。
OBJ_CREATEDBY	各オブジェクトを作成したエンティティを記録します。このロールを持つオブジェクトについては、C-38 ページの「 OBJ_CREATEDBY 変数 」を参照してください。
OBJ_STATE	登録されている各オブジェクトの現在の状態を記録します。このロールを持つオブジェクトについては、C-38 ページの「 OBJ_STATE 変数 」を参照してください。
VERSION	ワークスペースを管理しているスタンダード・フォームのバージョン番号を記録します。このロールを持つオブジェクトについては、C-39 ページの「 VERSION 変数 」を参照してください。
VISIBLE	Oracle OLAP の有効化ユーティリティで特定のオブジェクトをユーザーに表示するかどうかを示します。このロールを持つオブジェクトについては、C-36 ページの「 VISIBLE 変数 」を参照してください。

拡張クラス・オブジェクトのロール・プロパティ値

AW\$ROLE プロパティは、ワークスペース・オブジェクトによって実行される機能（ロール）を示します。拡張クラスのオブジェクトの場合、ロールは、DBMS_AWM やイネーブラなどの Oracle OLAP ユーティリティで内部的に使用されます。

DBA およびユーザーは、拡張クラスのオブジェクトについて、作成、変更または依存しないでください。このクラスのオブジェクトの、AW\$ROLE プロパティおよびすべてのプロパティは、独自の使用に限定されています。これらのオブジェクトのロールおよび関係のメンテナンスについて、Oracle では保証しません。

用語：ロール名を使用したオブジェクトの説明

スタンダード・フォーム規則には、ワークスペース・オブジェクトの名前に関する規則がないため、マニュアルでは名前でもオブジェクトを参照することができません。かわりに、オブジェクトの `AW$ROLE` プロパティの値をディスクリプタとして使用して、オブジェクトを説明しています。

たとえば、`cubedef` ディメンション、`aw_names` 変数および `default_hier` リレーションを参照するとします。この場合、`AW$ROLE` プロパティが `CUBEDEF`、`AW_NAMES` および `DEFAULT_HIER` に設定されているワークスペース・オブジェクトが参照先になっています。ほとんどのクラスでは、ワークスペース・オブジェクトの実際の名前は通常、ロールとほぼ同じですが、完全に一致しているわけではありません。

後続の項では、スタンダード・フォーム規則に準拠するロールを持つ各オブジェクトについて説明します。

実装クラスオブジェクト

実装クラスオブジェクトは、特定のワークスペースの論理オブジェクトに実装を提供します。一般的に、このオブジェクトは、ディメンションおよびメジャーとしてユーザーが認識するデータを保持します。実装クラスオブジェクトは、ワークスペースごとに異なります。たとえば、あるワークスペースでは `SALES` と `COST` というメジャーを保持し、別のワークスペースでは `BUDGET` と `ACTUAL` というメジャーを保持する場合があります。

`cubedef`、`measuredef` および `dimdef` オブジェクトは、それぞれキューブ、メジャーおよびディメンションを実装します。また、これらの各オブジェクトは、実装クラスのヘルパー・オブジェクトを持ちます。各オブジェクトの概要については、C-3 ページの「[論理モデルとワークスペース・オブジェクト](#)」を参照してください。

この項の後続の部分では、実装クラスの各オブジェクトについて説明します。この項の例では、スタンダード・フォームに必要なプロパティを示します。`Analytic Workspace Manager` または `DBMS_AWM` パッケージで作成されたワークスペースを調べる場合、様々なオブジェクトで追加のプロパティを確認する必要がある場合があります。追加のプロパティは、スタンダード・フォームへの準拠で必須ではありません。

プロパティに割り当てる必要がある値については、[表 C-1「実装クラスのプロパティ」](#) および [表 C-2「システム・プロパティ」](#) を参照してください。

特定のロールを持つすべてのオブジェクトを表示するには、`NAME` ディメンションをそのロールを持つすべてのオブジェクトに制限して、`NAME` ディメンションの値をレポートします。たとえば、`cubedef` オブジェクトをすべて表示するには、次の OLAP DML コマンドを実行します。

```
LIMIT name TO OBJ(PROPERTY 'AW$ROLE') EQ 'CUBEDEF'  
REPORT name
```

```
NAME
```

```
-----
```

```
UNITS_CUBE
PRICE_CUBE
```

キューブのオブジェクト

キューブは、*cubedef* デイメンションで実装されます。キューブは *loopspec* コンポジットも持ちます。

Cubedef デイメンション

論理キューブは、*AW\$ROLE* プロパティの値が *CUBEDEF* に指定されている、ワークスペース・デイメンションによって実装されます。各 *cubedef* デイメンションの値は、キューブの論理デイメンションの名前です。

cubedef デイメンションは親を持たないため、*AW\$PARENT_NAME* プロパティは *NA* に設定されます。論理キューブは、それに属しているメジャーの親です。

次に、*UNITS_CUBE* という名前の *cubedef* デイメンションの完全な記述を示します。

```
FULLDSC units_cube

DEFINE UNITS_CUBE DIMENSION TEXT
LD IMPLEMENTATION UNITS_CUBE Cube
PROPERTY 'AGGMAPLIST' 'GLOBAL_AW.GLOBAL!UNITS_CUBE_AGGMAP_AWCREATEDDEFAULT_1'
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:28:35'
PROPERTY 'AW$LOADPRGS' 'GLOBAL_AW.GLOBAL!__GET.CUBE.DATA_UNITS_CUBE_1'
PROPERTY 'AW$LOGICAL_NAME' 'UNITS_CUBE'
PROPERTY 'AW$LOOPSPEC' 'GLOBAL_AW.GLOBAL!UNITS_CUBE_COMPOSITE'
PROPERTY 'AW$PARENT_NAME' NA
PROPERTY 'AW$ROLE' 'CUBEDEF'
```

次に、*UNITS_CUBE* デイメンションの値を表示するレポートを示します。値は、キューブの論理デイメンションを実装する *dimdef* デイメンションの名前です。

```
REPORT units_cube

UNITS_CUBE
-----
CHANNEL
CUSTOMER
PRODUCT
TIME
```

Loopspec コンポジット

論理キューブは、*loopspec* コンポジットを持ちます。これを使用すると、キューブのメジャーについてデータ・アクセスの効率性が向上します。*loopspec* コンポジットは、スパー
スなデータを使用した繰り返し処理が必要な場合に特に有効です。

通常、*loopspec* コンポジットには、時間ディメンションを除く（存在する場合）、キューブの
すべてのディメンションが含まれます。*loopspec* の親は、サポート対象の論理キューブです。

次に、UNITS_CUBE という名前の論理キューブの *loopspec* コンポジットの完全な記述を示し
ます。コンポジットには、TIME ディメンションを除く、*cubedef*ディメンションの値として
リストされるすべてのディメンションが含まれています。

```
FULLDSC units_cube_composite

DEFINE UNITS_CUBE_COMPOSITE COMPOSITE <CUSTOMER PRODUCT CHANNEL>
LD IMPLEMENTATION Composite for UNITS_CUBE cube
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:28:40'
PROPERTY 'AW$PARENT_NAME' 'UNITS_CUBE'
PROPERTY 'AW$ROLE' 'LOOPSPEC'
PROPERTY 'AW$STATE' 'CREATED'
```

次に、この *loopspec* コンポジットの最初の 9 つの値を表示するレポートを示します。

```
REPORT units_cube_composite

CUSTOMER  PRODUCT  CHANNEL
-----
51         13       2
51         14       2
51         15       2
51         16       2
65         17       2
65         18       2
65         19       2
65         20       2
61         20       2
.          .        .
.          .        .
.          .        .
```

メジャーのオブジェクト

メジャーは、*measuredef* オブジェクトで実装されます。メジャーは、メジャーの集計規則を提供する *compspec aggmap* も持ちます。

Measuredef オブジェクト

論理メジャーは、*AW\$ROLE* プロパティの値が *MEASUREDEF* に指定されているワークスペース・オブジェクトによって実装されます。*measuredef* オブジェクトは、変数、式またはリレーションです。

measuredef オブジェクトの値は論理メジャーの値であり、親は論理キューブです。

次に、*UNITS* という名前の論理メジャーの *measuredef* オブジェクトの完全な記述を示します。オブジェクトは、親キューブ (*UNITS_CUBE*) のディメンションによってディメンション化されている式です。式には完全修飾されたオブジェクト名が含まれていますが、このタイプの指定はオプションです。

FULLDSC units

```
DEFINE UNITS FORMULA DECIMAL <TIME CUSTOMER PRODUCT CHANNEL>
LD IMPLEMENTATION UNITS Measure in UNITS_CUBE Cube
EQ aggregate( GLOBAL_AW.GLOBAL!UNITS_VARIABLE using
GLOBAL_AW.GLOBAL!UNITS_CUBE_AGGMAP_AWCREATEDDEFAULT_1)
PROPERTY 'AW$CLASS' - 'IMPLEMENTATION'
PROPERTY 'AW$COMPSPEC' 'GLOBAL_AW.GLOBAL!UNITS_CUBE_AGGMAP_AWCREATEDDEFAULT_1'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:28:41'
PROPERTY 'AW$LOGICAL_NAME' 'UNITS'
PROPERTY 'AW$PARENT_NAME' 'UNITS_CUBE'
PROPERTY 'AW$ROLE' 'MEASUREDEF'
PROPERTY 'AW$STATE' 'CREATED'
```

AW\$COMPSPEC プロパティは、メジャーの *compspec aggmap* の名前を保持します。このプロパティ値が *NA* の場合、格納されていないメジャーの値は計算されず、*NA* になります。

COMPSPEC Aggmap

論理メジャーには、*compspec* オブジェクトを含めることができます。このオブジェクトは、集計の計算規則を指定する *aggmap* です。

次に、*UNITS* という名前の論理メジャーの *compspec aggmap* の完全な記述を示します。*aggmap* には完全修飾されたオブジェクト名が含まれていますが、このタイプの指定はオプションです。

FULLDSC units_cube_aggmap_awcreateddefault_1

```
DEFINE UNITS_CUBE_AGGMAP_AWCREATEDDEFAULT_1 AGGMAP
```

```
LD IMPLEMENTATION Default aggmmap created by dbms_awm.refresh_awcube for UNITS_CUBE
cube
AGGMAP
RELATION GLOBAL_AW.GLOBAL!CHANNEL_PARENTREL OPERATOR SUM PRECOMPUTE(NA)
RELATION GLOBAL_AW.GLOBAL!CUSTOMER_PARENTREL OPERATOR SUM PRECOMPUTE(NA)
RELATION GLOBAL_AW.GLOBAL!PRODUCT_PARENTREL OPERATOR SUM PRECOMPUTE(NA)
RELATION GLOBAL_AW.GLOBAL!TIME_PARENTREL OPERATOR SUM PRECOMPUTE(NA)
AGGINDEX NO
END
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:29:24'
PROPERTY 'AW$PARENT_NAME' 'UNITS_CUBE'
PROPERTY 'AW$ROLE' 'COMPSPEC'
PROPERTY 'AW$STATE' 'CREATED'
```

ディメンションオブジェクト

ディメンションは、*dimdef* オブジェクトで実装されます。また、ディメンションは、次のサポート・オブジェクトを1つずつ持ちます。

- *Hierlist* ディメンション
- *Levellist* ディメンション
- *Member_levelrel* リレーション
- *Member_parentrel* リレーション
- *Hier_levels* 値セット

必要に応じて、ディメンションに1つ以上の *attrdef* オブジェクトを含めることもできます。

これらの各オブジェクトでは、*AW\$ROLE* プロパティによってオブジェクトのファンクションが記録されます。たとえば、*hierlist* ディメンションの *AW\$ROLE* プロパティは、*HIERLIST* に設定されます。また、これらの各オブジェクトの *AW\$PARENT* プロパティは、オブジェクトの所属先の論理ディメンションの名前を格納します。

ディメンションに階層、レベルまたはその両方がない場合、これらのサポート・オブジェクトは存在しますが、移入されません。

Dimdef ディメンション

論理ディメンションは、*AW\$ROLE* プロパティの値が *DIMDEF* に指定されているワークスペース・ディメンションによって実装されます。特定の *dimdef* ディメンションの値は、論理ディメンションの値です。

dimdef ディメンションは親を持たないため、*AW\$PARENT_NAME* プロパティは *NA* に設定されます。*AW\$TYPE* プロパティは、時間ディメンションの場合は *TIME* に、その他のディメンションの場合は *NA* に設定されます。

次に、PRODUCT という名前の論理ディメンションの *dimdef* ディメンションの完全な記述を示します。

```
FULLDSC product

DEFINE PRODUCT DIMENSION TEXT
LD IMPLEMENTATION PRODUCT Dimension
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:42'
PROPERTY 'AW$LOGICAL_NAME' 'PRODUCT'
PROPERTY 'AW$PARENT_NAME' NA
PROPERTY 'AW$ROLE' 'DIMDEF'
PROPERTY 'AW$STATE' 'ACTIVE'
PROPERTY 'AW$TYPE' NA
```

次に、すべてのレベルの *dimdef* ディメンションのサンプル値を表示するレポートを示します。これは、**埋込み合計**ディメンションです。次の例では、サロゲート・キーを使用することにより、すべてのレベルで値が重複しないようにしています。サロゲート・キーを使用しない場合は、別の方法で一意性を保証する必要があります。たとえば、接頭辞（ITEM.46 や FAMILY.7 など）としてレベルを使用する方法があります。例では、結果を明確に示すために、*attrdef* 変数および *member_levelrel* リレーションを含めています。

```
LIMIT product TO '46'
LIMIT product ADD ANCESTORS USING product_parentrel
REPORT DOWN product W 25 <product_long_description product_levelrel>
```

```
ALL_LANGUAGES: AMERICAN_AMERICA
```

-----PRODUCT_HIERLIST-----		
-----PRODUCT_ROLLUP-----		
PRODUCT	PRODUCT_LONG_DESCRIPTION	PRODUCT_LEVELREL

46	Standard Mouse	ITEM
7	Accessories	FAMILY
3	Software/Other	CLASS
1	Total Product	TOTAL_PRODUCT

Hierlist ディメンション

hierlist ディメンションは、親ディメンションの階層の名前を示します。つまり、*hierlist* ディメンションの値は、階層の名前（時間ディメンションの階層 CALENDAR および FISCAL など）です。階層はワークスペース・オブジェクトとして 1 対 1 の関係で実装されることがないため、名前はワークスペース・オブジェクトではなく論理階層を参照します。

次に、TIME_HIERLIST という名前の *hierlist* ディメンションの完全な記述を示します。

```
DEFINE TIME_HIERLIST DIMENSION TEXT
LD IMPLEMENTATION List of Hierarchies for TIME
```

```
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'HIERLIST'
PROPERTY 'AW$STATE' 'CREATED'
```

次に、この *hierlist* ディメンションの値を表示するレポートを示します。TIME は、CALENDAR という名前の階層を 1 つ持ちます。

```
REPORT time_hierlist
```

```
TIME_HIERLIST
-----
CALENDAR
```

Levellist ディメンション

levellist ディメンションは、親ディメンションのレベルの名前を示します。つまり、*levellist* ディメンションの値は、レベルの名前（地理ディメンションのレベル CITY、STATE、COUNTRY など）です。レベルはワークスペース・オブジェクトとして 1 対 1 の関係で実装されることがないため、名前はワークスペース・オブジェクトではなく論理レベルを参照します。各ディメンション値の論理レベルは、ディメンションの MEMBER_LEVELREL リレーションで識別されます。

次に、TIME_LEVELLIST という名前の *levellist* ディメンションの完全な記述を示します。

```
FULLDSC time_levellist

DEFINE TIME_LEVELLIST DIMENSION TEXT
LD IMPLEMENTATION List of levels for TIME
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$LOGICAL_NAME' NA
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'LEVELLIST'
PROPERTY 'AW$STATE' 'CREATED'
```

次に、この *levellist* ディメンションの値を表示するレポートを示します。レベルは、YEAR、QUARTER および MONTH です。

```
REPORT time_levellist
```

```
TIME_LEVELLIST
-----
YEAR
QUARTER
```


MONTH

Member_Levelrel リレーション

member_levelrel リレーションは、リレーションの親ディメンションの値ごとにレベルを記録します。たとえば、地理ディメンションの場合、member_levelrel リレーションは、BOSTON が CITY レベルに属し、IOWA が STATE レベルに属するというファクトを記録します。

次に、TIME_LEVELREL という名前の member_levelrel リレーションの完全な記述を示します。

```
FULLDSC time_levelrel

DEFINE TIME_LEVELREL RELATION TIME_LEVELLIST <TIME>
LD IMPLEMENTATION Level of each dimension member for TIME
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'MEMBER_LEVELREL'
PROPERTY 'AW$STATE' 'CREATED'
```

次に、この member_levelrel リレーションのサンプル値を表示するレポートを示します。レベルは、MONTH、QUARTER および YEAR です。

```
LIMIT time TO '75'
LIMIT time ADD ANCESTORS USING time_parentrel
REPORT DOWN time W 15 time_levelrel
```

TIME	TIME_LEVELREL
-----	-----
75	MONTH
83	QUARTER
85	YEAR

Member_Parentrel リレーション

member_parentrel リレーションは、リレーションの親ディメンションの値ごとに親ディメンションの値を記録します。たとえば、地理ディメンションの場合、member_parentrel リレーションは、BOSTON の親が MASSACHUSETTS であり、IOWA の親が USA であるというファクトを記録します。

次に、TIME_PARENTREL という名前の member_parentrel リレーションの完全な記述を示します。

```
FULLDSC time_parentrel

DEFINE TIME_PARENTREL RELATION TIME <TIME TIME_HIERLIST>
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
```

```
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'MEMBER_PARENTREL'
PROPERTY 'AW$STATE' 'CREATED'
```

次に、この *member_parentrel* リレーシヨンの値を表示するレポートを示します。特定の値の親は、検討対象の階層によって異なる場合があります。

```
REPORT DOWN time W 20 time_parentrel
```

---TIME_PARENTREL---	
---TIME_HIERLIST---	
TIME	CALENDAR

75	83
83	85
85	NA

Hier_Levels 値セット

hier_levels 値セットは、親ディメンシヨンの各階層に含まれるレベルを示します。

次に、TIME_HIER_LEVELS という名前の *hier_levels* 値セットの完全な記述を示します。

```
FULLDSC time_hier_levels
```

```
DEFINE TIME_HIER_LEVELS VALUESET TIME_LEVELLIST <TIME_HIERLIST>
LD IMPLEMENTATION Ordered from Bottom to Top list of levels in a hierarchy for TIME
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'HIER_LEVELS'
PROPERTY 'AW$STATE' 'CREATED'
```

次のコマンドでは、この *hier_levels* 値セットで記録されたレベルのリストを階層ごとに表示します。

```
REPORT W 25 VALUES(time_hier_levels)
```

TIME_HIERLIST	VALUES (TIME_HIER_LEVELS)

CALENDAR	MONTH
	QUARTER
	YEAR

Attrdef オブジェクト

論理属性は、AW\$ROLE プロパティの値が *attrdef* に指定されているワークスペース・オブジェクトによって実装されます。*attrdef* オブジェクトは、変数、式またはリレーションです。*attrdef* オブジェクトの値は論理属性の値であり、親は所属先の論理ディメンションです。

AW\$TYPE プロパティは、Oracle OLAP が属性に対して特別な使用方法を持っているかどうかを示します。特別な使用方法を示すプロパティ値は、DEFAULT_ORDER、END_DATE、TIME_SPAN、MEMBER_LONG_DESCRIPTION、MEMBER_SHORT_DESCRIPTION、MEMBER_VISIBLE です。値が USER または NA の場合、属性は Oracle OLAP で特別な意味を持ちません。

attrdef オブジェクトは、親 *dimdef* ディメンションでディメンション化する必要があります。*hierlist* ディメンション、ALL_LANGUAGES ディメンション、またはその両方のディメンションでディメンション化することもできます。

次に、TIME_LONG_DESCRIPTION という名前の *attrdef* オブジェクトの完全な記述を示します。次の long description 属性は、変数として実装されます。

```
FULLDSC time_long_description

DEFINE TIME_LONG_DESCRIPTION VARIABLE TEXT <TIME TIME_HIERLIST ALL_LANGUAGES>
LD IMPLEMENTATION LONG_DESCRIPTION Attribute for TIME Dimension
PROPERTY 'AW$CLASS' 'IMPLEMENTATION'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:28:25'
PROPERTY 'AW$LOGICAL_NAME' 'LONG_DESCRIPTION'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'ATTRDEF'
PROPERTY 'AW$STATE' 'CREATED'
PROPERTY 'AW$TYPE' 'Long Description'
```

次に、この *attrdef* オブジェクトの値をレベルごとに選択して表示するレポートを示します。

```
LIMIT time TO time_levelrel EQ 'YEAR'
LIMIT time KEEP LAST 1
LIMIT time ADD DESCENDANTS USING time_parentrel
REPORT DOWN time W 25 time_long_description

ALL_LANGUAGES: AMERICAN_AMERICA
               --TIME_LONG_DESCRIPTION--
               -----TIME_HIERLIST-----
TIME           CALENDAR
-----
119            2004
115            Q1-04
116            Q2-04
103            Jan-04
```

104	Feb-04
105	Mar-04
106	Apr-04
107	May-04
108	Jun-04

カタログ・クラスオブジェクト

カタログ・クラスオブジェクトは、ワークスペース内の論理オブジェクトに関する情報を保持します。カタログ・クラスオブジェクトには、ワークスペース内のすべてのキューブのリスト、ワークスペース内のすべてのメジャーのリスト、ワークスペース内のすべてのディメンションのリスト、および様々なユーティリティの作業を容易にするその他のリストが含まれます。特定のワークスペースには、各カタログ・クラスオブジェクトのインスタンスが1つあります。DBMS_AWM は、ロールを名前として使用してこれらのオブジェクトを作成するため、*all_languages* ディメンションには ALL_LANGUAGES という名前が付きます。このため、カタログ・クラスオブジェクトの名前を、ここでは大文字で表記して実際の名前を示します。

この項では、カタログ・クラスオブジェクトを次のグループに分類して説明します。

- [オブジェクトのリスト](#)
- [タイプ、ロールおよび言語のリスト](#)
- [キューブおよびディメンションオブジェクトのリスト](#)
- [オブジェクト情報のサポート](#)

オブジェクトのリスト

カタログ・クラスにはディメンションのセットが含まれます。ディメンションの各セットは、特定の種類のオブジェクトをすべて示します。たとえば、ALL_MEASURES ディメンションの場合、すべての論理メジャーを示します。

ALL_CUBES ディメンション

ALL_CUBES ディメンションは、ワークスペース内のすべての論理キューブのフルネームを示します。次に、ALL_CUBES ディメンションの完全な記述を示します。

```
FULLDSC all_cubes

DEFINE ALL_CUBES DIMENSION TEXT
LD CATALOGS List of all cubes in the aw
PROPERTY 'AW$CLASS' 'CATALOGS'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'ALL_CUBES'
PROPERTY 'AW$STATE' 'CREATED'
```

次に、この ALL_CUBES ディメンションの値のレポートを示します。

```
REPORT W 30 all_cubes
```

```
ALL_CUBES
```

```
-----  
GLOBAL_AW.UNITS_CUBE.CUBE
```

```
GLOBAL_AW.PRICE_CUBE.CUBE
```

ALL_MEASURES ディメンション

ALL_MEASURES ディメンションは、ワークスペース内のすべての論理メジャーのフルネームフルネームを示します。

このディメンションの完全な記述は、C-24 ページの「[ALL_CUBES ディメンション](#)」の ALL_CUBES ディメンションで示したものとほぼ同じです。次に、ALL_MEASURES ディメンションの値のレポートを示します。

```
REPORT W 40 all_measures
```

```
ALL_MEASURES
```

```
-----  
GLOBAL_AW.UNITS_CUBE.UNITS.MEASURE
```

```
GLOBAL_AW.PRICE_CUBE.UNIT_COST.MEASURE
```

```
GLOBAL_AW.PRICE_CUBE.UNIT_PRICE.MEASURE
```

ALL_DIMENSIONS ディメンション

ALL_DIMENSIONS ディメンションは、ワークスペース内のすべての論理ディメンションのフルネームを示します。

このディメンションの完全な記述は、C-24 ページの「[ALL_CUBES ディメンション](#)」の ALL_CUBES ディメンションで示したものとほぼ同じです。次に、ALL_DIMENSIONS ディメンションの値のレポートを示します。

```
REPORT W 40 all_dimensions
```

```
ALL_DIMENSIONS
```

```
-----  
GLOBAL_AW.CHANNEL.DIMENSION
```

```
GLOBAL_AW.CUSTOMER.DIMENSION
```

```
GLOBAL_AW.PRODUCT.DIMENSION
```

```
GLOBAL_AW.TIME.DIMENSION
```

ALL_HIERARCHIES ディメンション

ALL_HIERARCHIES ディメンションは、ワークスペース内のすべての階層のフルネームを示します。

このディメンションの完全な記述は、C-24 ページの「[ALL_CUBES ディメンション](#)」の ALL_CUBES ディメンションで示したものとほぼ同じです。次に、ALL_HIERARCHIES ディメンションの値のレポートを示します。

```
REPORT W 45 all_hierarchies
```

```
ALL_HIERARCHIES
-----
GLOBAL_AW.CHANNEL.CHANNEL_ROLLUP.HIERARCHY
GLOBAL_AW.CHANNEL.AW$NONE.HIERARCHY
GLOBAL_AW.CUSTOMER.SHIPMENTS_ROLLUP.HIERARCHY
GLOBAL_AW.CUSTOMER.MARKET_ROLLUP.HIERARCHY
GLOBAL_AW.CUSTOMER.AW$NONE.HIERARCHY
GLOBAL_AW.PRODUCT.PRODUCT_ROLLUP.HIERARCHY
GLOBAL_AW.PRODUCT.AW$NONE.HIERARCHY
GLOBAL_AW.TIME.CALENDAR.HIERARCHY
GLOBAL_AW.TIME.AW$NONE.HIERARCHY
```

階層に AW\$NONE という単純名が付いている場合、ディメンションに階層がないことを示します。

ALL_LEVELS ディメンション

ALL_LEVELS ディメンションは、ワークスペース内のすべてのレベルのフルネームを示します。

このディメンションの完全な記述は、C-24 ページの「[ALL_CUBES ディメンション](#)」の ALL_CUBES ディメンションで示したものとほぼ同じです。次に、ALL_LEVELS ディメンションの値のレポートを示します。

```
REPORT W 40 all_levels
```

```
ALL_LEVELS
-----
GLOBAL_AW.CHANNEL.ALL_CHANNELS.LEVEL
GLOBAL_AW.CHANNEL.CHANNEL.LEVEL
GLOBAL_AW.CHANNEL.AW$NONE.LEVEL
GLOBAL_AW.CUSTOMER.ALL_CUSTOMERS.LEVEL
GLOBAL_AW.CUSTOMER.REGION.LEVEL
GLOBAL_AW.CUSTOMER.WAREHOUSE.LEVEL
GLOBAL_AW.CUSTOMER.TOTAL_MARKET.LEVEL
GLOBAL_AW.CUSTOMER.MARKET_SEGMENT.LEVEL
GLOBAL_AW.CUSTOMER.ACCOUNT.LEVEL
GLOBAL_AW.CUSTOMER.SHIP_TO.LEVEL
GLOBAL_AW.CUSTOMER.AW$NONE.LEVEL
GLOBAL_AW.PRODUCT.TOTAL_PRODUCT.LEVEL
GLOBAL_AW.PRODUCT.CLASS.LEVEL
GLOBAL_AW.PRODUCT.FAMILY.LEVEL
```

```
GLOBAL_AW.PRODUCT.ITEM.LEVEL
GLOBAL_AW.PRODUCT.AW$NONE.LEVEL
GLOBAL_AW.TIME.YEAR.LEVEL
GLOBAL_AW.TIME.QUARTER.LEVEL
GLOBAL_AW.TIME.MONTH.LEVEL
GLOBAL_AW.TIME.AW$NONE.LEVEL
```

ALL_ATTRIBUTES ディメンション

ALL_ATTRIBUTES ディメンションは、ワークスペース内のすべての属性のフルネームを示します。

このディメンションの完全な記述は、C-24 ページの「[ALL_CUBES ディメンション](#)」の ALL_CUBES ディメンションで示したものとほぼ同じです。次に、ALL_ATTRIBUTES ディメンションの値のレポートを示します。

```
REPORT W 50 all_attributes
```

```
ALL_ATTRIBUTES
```

```
-----
GLOBAL_AW.CHANNEL.LONG_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.CHANNEL.SHORT_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.CUSTOMER.LONG_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.CUSTOMER.SHORT_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.PRODUCT.LONG_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.PRODUCT.SHORT_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.PRODUCT.PACKAGE.ATTRIBUTE
GLOBAL_AW.TIME.LONG_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.TIME.SHORT_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.TIME.END_DATE.ATTRIBUTE
GLOBAL_AW.TIME.TIME_SPAN.ATTRIBUTE
```

ALL_OBJECTS ディメンション

ALL_OBJECTS ディメンションは、ワークスペース内のすべての論理オブジェクトのフルネームを示します。

次に、ALL_OBJECTS ディメンションの完全な記述を示します。

```
FULLDSC all_objects
```

```
DEFINE ALL_OBJECTS DIMENSION CONCAT (ALL_DIMENSIONS ALL_CUBES ALL_MEASURES
ALL_HIERARCHIES ALL_LEVELS ALL_ATTRIBUTES)
LD CATALOGS List of all objects in the aw
PROPERTY 'AW$CLASS' 'CATALOGS'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:07:35'
PROPERTY 'AW$ROLE' 'ALL_OBJECTS'
PROPERTY 'AW$STATE' 'CREATED'
```

ALL_OBJECTS は、ALL_CUBES、ALL_MEASURES、ALL_HIERARCHIES、ALL_LEVELS および ALL_ATTRIBUTES ディメンションの CONCAT ディメンションです。ディメンション・メンバーは、これらのディメンションのメンバーの連結リストです（次の例を参照）。

```

LIMIT all_cubes TO FIRST 2
LIMIT all_measures TO FIRST 2
LIMIT all_hierarchies TO FIRST 2
LIMIT all_levels TO FIRST 2
LIMIT all_attributes TO FIRST 2
LIMIT all_objects TO all_cubes
LIMIT all_objects ADD all_measures
LIMIT all_objects ADD all_hierarchies
LIMIT all_objects ADD all_levels
LIMIT all_objects ADD all_attributes
REPORT W 70 all_objects

ALL_OBJECTS
-----
<ALL_CUBES: GLOBAL_AW.UNITS_CUBE.CUBE>
<ALL_CUBES: GLOBAL_AW.PRICE_CUBE.CUBE>
<ALL_MEASURES: GLOBAL_AW.UNITS_CUBE.UNITS.MEASURE>
<ALL_MEASURES: GLOBAL_AW.PRICE_CUBE.UNIT_COST.MEASURE>
<ALL_HIERARCHIES: GLOBAL_AW.CHANNEL.CHANNEL_ROLLUP.HIERARCHY>
<ALL_HIERARCHIES: GLOBAL_AW.CHANNEL.AW$NONE.HIERARCHY>
<ALL_LEVELS: GLOBAL_AW.CHANNEL.ALL_CHANNELS.LEVEL>
<ALL_LEVELS: GLOBAL_AW.CHANNEL.CHANNEL.LEVEL>
<ALL_ATTRIBUTES: GLOBAL_AW.CHANNEL.LONG_DESCRIPTION.ATTRIBUTE>
<ALL_ATTRIBUTES: GLOBAL_AW.CHANNEL.SHORT_DESCRIPTION.ATTRIBUTE>

```

タイプ、ロールおよび言語のリスト

カタログ・クラスには、現行のバージョンのスタンダード・フォームでサポートされているタイプおよびロールを示すディメンションが含まれます。また、現行のアナリティック・ワークスペースでサポートされている言語を示すディメンションもあります。

ALL_OBJTYPES ディメンション

ALL_OBJTYPES ディメンションは、現行のバージョンのスタンダード・フォームでサポートされているすべてのオブジェクト型を示します。次に、型を表示するレポートを示します。

```

REPORT all_objtypes

ALL_OBJTYPES
-----
CUBE

```


MEASURE
 DIMENSION
 LEVEL
 HIERARCHY
 ATTRIBUTE

ALL_DESCTYPES ディメンション

ALL_DESCTYPES ディメンションは、現行のバージョンのスタンダード・フォームで認識されるすべての説明のタイプを示します。次に、タイプを表示するレポートを示します。

REPORT all_descypes

ALL_DESCTYPES

 SHORT
 LONG
 PLURAL

ALL_ATTRTYPES ディメンション

ALL_ATTRTYPES ディメンションは、現行のバージョンのスタンダード・フォームで認識されるすべての属性のタイプを示します。次に、タイプを表示するレポートを示します。

REPORT W 40 all_attrtypes

ALL_ATTRTYPES

 DEFAULT_ORDER
 END_DATE
 TIME_SPAN
 MEMBER_LONG_DESCRIPTION
 MEMBER_SHORT_DESCRIPTION
 MEMBER_VISIBLE
 USER

AW_ROLES ディメンション

AW_ROLES ディメンションは、現行のバージョンのスタンダード・フォームで認識されるすべてのロールを示します。次に、ロールを表示するレポートを示します。

REPORT W 30 aw_roles

AW_ROLES

 LANGUAGEDEF
 ADVIEWLIST
 ADTLIST

ADTTBLLIST
ADTREL
ADTTBLREL
ADTLMIMAP
DIMDEF
MEMBER_CREATEDBY
LEVELLIST
MEMBER_LEVELREL
LEVEL_CREATEDBY
LEVELCOLLIST
LEVELCOLNUM
LEVELCOLMAP
HIERLIST
HIER_CREATEDBY
MEMBER_INHIER
MEMBER_PARENTREL
ATTRDEF
SRCCOMPOSITE
SRCLVOWNER
SRCLVLTBL
SRCLVLCOL
SRCLVLENTCOL
MEMBER_FAMILYREL
HIER_LEVELS
MEMBER_GID
ALL_LANGUAGES
ALL_DIMENSIONS
ALL_CUBES
ALL_MEASURES
ALL_HIERARCHIES
ALL_LEVELS
ALL_ATTRIBUTES
AW_ROLES
ALL_DESCTYPES
ALL_OBJTYPES
ALL_OBJECTS
AW_NAMES
AW_COMPSPECS
AW_LOOP SPECS

ALL_LANGUAGES ディメンション

ALL_LANGUAGES ディメンションは、現行のアナリティック・ワークスペースで実装されるすべての言語を示します。次のレポートでは、サンプル・ワークスペースで実装される言語を1つ表示します。言語名は、グローバル化・サポート規格に準拠する必要があります。

```
REPORT W 30 all_languages
```

```
ALL_LANGUAGES
```

```
-----  
AMERICAN_AMERICA
```

キューブおよびディメンションのオブジェクトのリスト

カタログ・クラスには、各キューブのメジャーに加え、各ディメンションの階層、レベルおよび属性を示す値セットが含まれます。これらのリストは、特定のワークスペースに固有です。

CUBE_MEASURES 値セット

CUBE_MEASURES 値セットは、現行のアナリティック・ワークスペースの各キューブに属しているメジャーを示します。値セットは、ALL_CUBES によってディメンション化されるため、各キューブは独自のリストを持ちます。次に、サンプル・ワークスペースの CUBE_MEASURES 値セットの完全な記述を示します。

```
FULLDSC cube_measures
```

```
DEFINE CUBE_MEASURES VALUESET ALL_MEASURES <ALL_CUBES>  
PROPERTY 'AW$CLASS' 'CATALOGS'  
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'  
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'  
PROPERTY 'AW$ROLE' 'CUBE_MEASURES'  
PROPERTY 'AW$STATE' 'CREATED'
```

次のコマンドでは、各キューブに関連付けられているメジャーのリストを表示します。

```
LCOLWIDTH=30 "Widen the label column"  
REPORT W 40 VALUES(cube_measures)
```

```
ALL_CUBES                                VALUES(CUBE_MEASURES)  
-----  
GLOBAL_AW.UNITS_CUBE.CUBE               GLOBAL_AW.UNITS_CUBE.UNITS.MEASURE  
GLOBAL_AW.PRICE_CUBE.CUBE                GLOBAL_AW.PRICE_CUBE.UNIT_COST.MEASURE  
                                           GLOBAL_AW.PRICE_CUBE.UNIT_PRICE.MEASURE
```

DIM_HIERARCHIES 値セット

DIM_HIERARCHIES 値セットは、現行のアナリティック・ワークスペースの各ディメンションに属している階層を示します。値セットは、ALL_DIMENSIONS によってディメンション化されるため、各ディメンションは独自のリストを持ちます。次のコマンドでは、各ディメンションの階層のリストを表示します。

```
REPORT W 45 VALUES(dim_hierarchies)
```

ALL_DIMENSIONS	VALUES (DIM_HIERARCHIES)
-----	-----
GLOBAL_AW.CHANNEL.DIMENSION	GLOBAL_AW.CHANNEL.CHANNEL_ROLLUP.HIERARCHY
GLOBAL_AW.CUSTOMER.DIMENSION	GLOBAL_AW.CUSTOMER.SHIPMENTS_ROLLUP.HIERARCHY
	GLOBAL_AW.CUSTOMER.MARKET_ROLLUP.HIERARCHY
GLOBAL_AW.PRODUCT.DIMENSION	GLOBAL_AW.PRODUCT.PRODUCT_ROLLUP.HIERARCHY
GLOBAL_AW.TIME.DIMENSION	GLOBAL_AW.TIME.CALENDAR.HIERARCHY

DIM_LEVELS 値セット

DIM_LEVELS 値セットは、現行のアナリティック・ワークスペースの各ディメンションに属しているレベルを示します。値セットは、ALL_DIMENSIONS によってディメンション化されるため、各ディメンションは独自のリストを持ちます。次のコマンドでは、各ディメンションのレベルのリストを表示します。

REPORT W 45 VALUES (dim_levels)

ALL_DIMENSIONS	VALUES (DIM_LEVELS)
-----	-----
GLOBAL_AW.CHANNEL.DIMENSION	GLOBAL_AW.CHANNEL.ALL_CHANNELS.LEVEL
	GLOBAL_AW.CHANNEL.CHANNEL.LEVEL
GLOBAL_AW.CUSTOMER.DIMENSION	GLOBAL_AW.CUSTOMER.ALL_CUSTOMERS.LEVEL
	GLOBAL_AW.CUSTOMER.REGION.LEVEL
	GLOBAL_AW.CUSTOMER.WAREHOUSE.LEVEL
	GLOBAL_AW.CUSTOMER.TOTAL_MARKET.LEVEL
	GLOBAL_AW.CUSTOMER.MARKET_SEGMENT.LEVEL
	GLOBAL_AW.CUSTOMER.ACCOUNT.LEVEL
	GLOBAL_AW.CUSTOMER.SHIP_TO.LEVEL
GLOBAL_AW.PRODUCT.DIMENSION	GLOBAL_AW.PRODUCT.TOTAL_PRODUCT.LEVEL
	GLOBAL_AW.PRODUCT.CLASS.LEVEL
	GLOBAL_AW.PRODUCT.FAMILY.LEVEL
	GLOBAL_AW.PRODUCT.ITEM.LEVEL
GLOBAL_AW.TIME.DIMENSION	GLOBAL_AW.TIME.YEAR.LEVEL
	GLOBAL_AW.TIME.QUARTER.LEVEL
	GLOBAL_AW.TIME.MONTH.LEVEL

DIM_ATTRIBUTES 値セット

DIM_ATTRIBUTES 値セットは、現行のアナリティック・ワークスペースの各ディメンションに属している属性を示します。値セットは、ALL_DIMENSIONS によってディメンション化されるため、各ディメンションは独自のリストを持ちます。次のコマンドでは、TIME ディメンションの属性のリストを表示します。

REPORT W 46 VALUES (dim_attributes)

ALL_DIMENSIONS	VALUES (DIM_ATTRIBUTES)
-----	-----
GLOBAL_AW.CHANNEL.DIMENSION	GLOBAL_AW.CHANNEL.LONG_DESCRIPTION.ATTRIBUTE

GLOBAL_AW.CUSTOMER.DIMENSION	GLOBAL_AW.CHANNEL.SHORT_DESCRIPTION.ATTRIBUTE GLOBAL_AW.CUSTOMER.LONG_DESCRIPTION.ATTRIBUTE GLOBAL_AW.CUSTOMER.SHORT_DESCRIPTION.ATTRIBUTE
GLOBAL_AW.PRODUCT.DIMENSION	GLOBAL_AW.PRODUCT.LONG_DESCRIPTION.ATTRIBUTE GLOBAL_AW.PRODUCT.SHORT_DESCRIPTION.ATTRIBUTE GLOBAL_AW.PRODUCT.PACKAGE.ATTRIBUTE
GLOBAL_AW.TIME.DIMENSION	GLOBAL_AW.TIME.LONG_DESCRIPTION.ATTRIBUTE GLOBAL_AW.TIME.SHORT_DESCRIPTION.ATTRIBUTE GLOBAL_AW.TIME.END_DATE.ATTRIBUTE GLOBAL_AW.TIME.TIME_SPAN.ATTRIBUTE

オブジェクト情報のサポート

カタログ・クラスには、他の様々なオブジェクトをサポートするオブジェクトを示す変数および式が含まれます。

AW_NAMES 変数

AW_NAMES 変数は、ALL_OBJECTS によってディメンション化されます。この変数には、各論理オブジェクトを実装するワークスペース・オブジェクトの名前が格納されます。特定の論理オブジェクトを実装するワークスペース・オブジェクトがない場合、値は NA です。

次に、AW_NAMES 変数の完全な記述を示します。

```
FULLDSC aw_names

DEFINE AW_NAMES VARIABLE TEXT <ALL_OBJECTS>
PROPERTY 'AW$CLASS' 'CATALOGS'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'AW_NAMES'
PROPERTY 'AW$STATE' 'CREATED'
```

AW_COMPSPECS 変数

AW_COMPSPECS 変数は、ALL_DIMENSIONS によってディメンション化されます。各論理ディメンションでは、AW_COMPSPECS 変数に、ディメンションの変更時に変更する必要がある AGGMAP オブジェクトの名前が格納されます。

次に、AW_COMPSPECS 変数の完全な記述を示します。

```
FULLDSC aw_comspecs

DEFINE AW_COMPSPECS VARIABLE TEXT <ALL_DIMENSIONS>
PROPERTY 'AW$CLASS' 'CATALOGS'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'AW_COMPSPECS'
```

```
PROPERTY 'AW$STATE' 'CREATED'
```

AW_LOOPSPECS 変数

AW_LOOPSPECS 変数は、ALL_CUBES によってディメンション化されます。この変数には、各キューブのワークスペース・コンポジットの名前が格納されます。

次に、AW_LOOPSPECS 変数の完全な記述を示します。

```
FULLDSC aw_loopspecs
```

```
DEFINE AW_LOOPSPECS VARIABLE TEXT <ALL_CUBES>
PROPERTY 'AW$CLASS' 'CATALOGS'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'AW_LOOPSPECS'
PROPERTY 'AW$STATE' 'CREATED'
```

機能クラスオブジェクト

機能クラスオブジェクトは、特定の論理オブジェクトおよびそれを実装するワークスペース・オブジェクトに関する情報を保持します。たとえば、すべての論理オブジェクトの説明を格納するオブジェクトや、オブジェクトをユーザーに表示するかどうかを示すオブジェクトなどがあります。

ALL_DESCRIPTIONS 変数

ALL_DESCRIPTIONS 変数は、様々な論理オブジェクトの簡単な説明、詳細な説明および複数説明を格納します。参照を容易にするために、この変数はコンポジットでディメンション化されています。

次に、ALL_DESCRIPTIONS 変数の完全な記述を示します。

```
FULLDSC all_descriptions
```

```
DEFINE ALL_DESCRIPTIONS VARIABLE TEXT <SPARSE <ALL_OBJECTS ALL_DESCTYPES ALL_LANGUAGES>>
LD FEATURES Descriptions for all objects
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'ALL_DESCRIPTIONS'
PROPERTY 'AW$STATE' 'CREATED'
```

次のレポートでは、ALL_DESCRIPTIONS のサンプルの値を示します。

```
report w 30 down all_dimensions all_descriptions
```

```
ALL_LANGUAGES: AMERICAN_AMERICA
```

ALL_DIMENSIONS	-----ALL_DESCRIPTIONS-----		
	-----ALL_DESCTYPES-----		
	SHORT	LONG	PLURAL
GLOBAL_AW.CHANNEL.DIMENSION	Channel	NA	CHANNEL
GLOBAL_AW.CUSTOMER.DIMENSION	Customer	NA	CUSTOMER
GLOBAL_AW.PRODUCT.DIMENSION	Product	NA	PRODUCT
GLOBAL_AW.TIME.DIMENSION	Time	NA	TIME

ATTR_INHIER 変数

ATTR_INHIER 変数は、特定の属性が特定の階層に関連付けられているかどうかを示すブール変数です。変数は、ALL_ATTRIBUTES および ALL_HIERARCHIES によってディメンション化されます。

次に、ATTR_INHIER 変数の完全な記述を示します。

```
FULLDSC attr_inhier

DEFINE ATTR_INHIER VARIABLE BOOLEAN <ALL_ATTRIBUTES ALL_HIERARCHIES>
LD FEATURES Indicates if each attribute participates in each hierarchy
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'ATTR_INHIER'
PROPERTY 'AW$STATE' 'CREATED'
```

DEFAULT_HIER リレーション

DEFAULT_HIER リレーションは、各ディメンションのデフォルトの階層のフルネームを記録します。リレーションのベース・ディメンションは、ALL_DIMENSIONS です。

次に、DEFAULT_HIER リレーションの完全な記述を示します。

```
FULLDSC default_hier

DEFINE DEFAULT_HIER RELATION ALL_HIERARCHIES <ALL_DIMENSIONS>
LD FEATURES Default hierarchy for each dimension
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'DEFAULT_HIER'
PROPERTY 'AW$STATE' 'CREATED'
```

次のレポートでは、各ディメンションのデフォルトの階層を示します。

```
REPORT W 45 default_hier
```

ALL_DIMENSIONS	DEFAULT_HIER
-----	-----
GLOBAL_AW.CHANNEL.DIMENSION	GLOBAL_AW.CHANNEL.CHANNEL_ROLLUP.HIERARCHY
GLOBAL_AW.CUSTOMER.DIMENSION	GLOBAL_AW.CUSTOMER.SHIPMENTS_ROLLUP.HIERARCHY
GLOBAL_AW.PRODUCT.DIMENSION	GLOBAL_AW.PRODUCT.PRODUCT_ROLLUP.HIERARCHY
GLOBAL_AW.TIME.DIMENSION	GLOBAL_AW.TIME.CALENDAR.HIERARCHY

VISIBLE 変数

VISIBLE 変数は、登録されているオブジェクトを Oracle OLAP イネーブラ・ユーティリティで表示するか無視するかを示すブール変数です。変数は、ALL_OBJECTS によってディメンション化されるため、各オブジェクトは独自の設定を持ちます。

次に、VISIBLE 変数の完全な記述を示します。

```
FULLDSC visible

DEFINE VISIBLE VARIABLE BOOLEAN <ALL_OBJECTS>
LD FEATURES Is the object visible
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'VISIBLE'
PROPERTY 'AW$STATE' 'CREATED'
```

Member_Inhier 変数

member_inhier 変数は、ディメンションの特定のメンバーが特定の階層内にあるかどうかを示すブール変数です。ワークスペース内の各ディメンションには、これらの変数のいずれか1つがあり、そのディメンションが変数の親になります。

次に、TIME ディメンションの *member_inhier* 変数の完全な記述を示します。

```
FULLDSC time_inhier

DEFINE TIME_INHIER VARIABLE BOOLEAN <TIME TIME_HIERLIST>
LD FEATURES Indicator of whether each dimension member participates in a hierarchy
for TIME
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'MEMBER_INHIER'
PROPERTY 'AW$STATE' 'CREATED'
```


Member_Createdby 変数

member_createdby 変数は、特定のディメンションの各メンバーを作成したエンティティを記録します。ワークスペース内の各ディメンションには、これらの変数のいずれか1つがあり、そのディメンションが変数の親になります。

次に、TIME ディメンションの *member_createdby* 変数の完全な記述を示します。

```
FULLDSC time_createdby

DEFINE TIME_CREATEDBY VARIABLE TEXT <TIME>
LD FEATURES Creator of each dimension member for TIME
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'MEMBER_CREATEDBY'
PROPERTY 'AW$STATE' 'CREATED'
```

Member_Familyrel リレーション

member_familyrel リレーションは、ディメンションの特定のメンバーの祖先を記録します。ワークスペース内の各ディメンションには、これらのリレーションのいずれか1つがあり、そのディメンションが変数の親になります。これらのリレーションは、内部的に使用されます。

次に、TIME ディメンションの *member_familyrel* リレーションの完全な記述を示します。

```
FULLDSC time_familyrel

DEFINE TIME_FAMILYREL RELATION TIME <TIME TIME_LEVELLIST TIME_HIERLIST>
LD FEATURES Family/Ancestry structure for TIME
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'MEMBER_FAMILYREL'
PROPERTY 'AW$STATE' 'CREATED'
```

Member_Gid 変数

member_gid 変数は、特定の階層内におけるディメンションの特定のメンバーのレベルの深さを記録します。ワークスペース内の各ディメンションには、これらのリレーションのいずれか1つがあり、そのディメンションが変数の親になります。これらのリレーションは、内部的に使用されます。

次に、TIME ディメンションの *member_gid* リレーションの完全な記述を示します。

```
FULLDSC time_gid
```

```
DEFINE TIME_GID VARIABLE INTEGER <TIME TIME_HIERLIST>
LD FEATURES Grouping id value for TIME
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '03SEP03_15:27:47'
PROPERTY 'AW$PARENT_NAME' 'TIME'
PROPERTY 'AW$ROLE' 'MEMBER_GID'
PROPERTY 'AW$STATE' 'CREATED'
```

OBJ_CREATEDBY 変数

OBJ_CREATEDBY 変数は、スタンダード・フォームに登録されている各オブジェクトを作成したエンティティを記録します。変数は、ALL_OBJECTS によってディメンション化されます。

次に、OBJ_CREATEDBY 変数の完全な記述を示します。

```
FULLDSC obj_createdby
```

```
DEFINE OBJ_CREATEDBY VARIABLE TEXT <ALL_OBJECTS>
LD FEATURES Creator of each object
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'OBJ_CREATEDBY'
PROPERTY 'AW$STATE' 'CREATED'
```

OBJ_STATE 変数

OBJ_STATE 変数は、スタンダード・フォームに登録されている各オブジェクトの状態を記録します。変数は、ALL_OBJECTS によってディメンション化されます。各オブジェクトの値は、UNDER_CONSTRUCTION または ACTIVE です。

次に、OBJ_STATE 変数の完全な記述を示します。

```
FULLDSC obj_state
```

```
DEFINE OBJ_STATE VARIABLE TEXT <ALL_OBJECTS>
LD FEATURES State of each object
PROPERTY 'AW$CLASS' 'FEATURES'
PROPERTY 'AW$CREATEDBY' 'AW$CREATE'
PROPERTY 'AW$LASTMODIFIED' '04DEC02_13:09:14'
PROPERTY 'AW$ROLE' 'OBJ_STATE'
PROPERTY 'AW$STATE' 'CREATED'
```

VERSION 変数

VERSION 変数は、アナリティック・ワークスペースが管理されているスタンダード・フォーム規則のバージョン番号を記録します。

拡張クラスのオブジェクト

拡張クラスのオブジェクトは、Oracle OLAP ユーティリティで定義およびメンテナンスされます。このオブジェクトは、スタンダード・フォームへの独自の拡張であり、リリース間でのメンテナンスについて Oracle では保証されません。

拡張クラスのオブジェクトは、定義、変更または依存しないください。

用語集

DBA

データベース管理者。Oracle Database の作成、インストール、構成およびメンテナンスの担当者。

EIF ファイル(EIF file)

アナリティック・ワークスペース間でデータを転送するための形式に準拠したファイル。EIF ファイルを作成する場合は OLAP DML の EXPORT コマンドを使用し、EIF ファイルを読み込む場合は IMPORT コマンドを使用する。

NA 値(NA value)

データが「使用不可 (NA)」であることを示す特殊なデータ値。特定のデータ値が割り当てられていないセル、またはデータを計算することができないセルの値は NA 値となる。

「[セル](#)」、「[スパース性](#)」も参照。

OLAP DML

アナリティック・ワークスペース用の低レベルのデータ定義および操作言語。

OLAP カタログ(OLAP Catalog)

データを多次元用語（キューブ、メジャー、ディメンション、属性など）で記述する一連の読み込み API および書き込み API で構成されたメタデータ・パッケージ。

「[メタデータ](#)」も参照。

QDR

「[修飾データ参照](#)」を参照。

値セット(valueset)

ワークスペース・オブジェクトの一種。値セットには、特定のディメンションのディメンション・メンバーのリストが格納される。値セットを定義したら、LIMIT コマンドを使用し

でディメンションのメンバーを値セットに割り当てる。値セットの値は、Oracle OLAP のセッションをまたがって保存できる。

新しい Oracle OLAP セッションを開始したり、ワークスペースを開始したりすると、各ディメンションのすべての値にステータスが設定される。ここで値セットを使用すると、ディメンションを、そのディメンションの値セットに格納されている値に制限できる。

「[ディメンション](#)」も参照。

アナリティック・ワークスペース(analytic workspace)

リレーショナル・データベースの LOB 表に格納される多次元スキーマ。アナリティック・ワークスペースには、様々なオブジェクトを格納できる。このようなオブジェクトの中には、他のオブジェクトと結合して一体化できるものもあるが、それ以外のオブジェクトは完全に独立している。アプリケーションにとって有用なデータを格納しているオブジェクトも、DBA または開発者が使用するためにのみ存在しているオブジェクトもある。オブジェクトには、多次元モデルの中で様々な役割を果たすいくつかの基本的なタイプがある。このような点で、アナリティック・ワークスペースはリレーショナル・スキーマとよく似ている。

アナリティック・ワークスペースで各種操作を行うための基本的な低レベルの言語として、OLAP DML がある。PL/SQL および Java に習熟しているユーザーは、OLAP DML へのインタフェースを提供する、これらの言語によるツールを使用できる。

「[OLAP DML](#)」も参照。

埋込み合計(embedded total)

階層が属するディメンションに組み込まれた、事前定義の集計レベル。たとえば、時間ディメンションでは、各四半期はその四半期に含まれる月の合計を表す。埋込み合計のデータは、アナリティック・ワークスペースで集計システムによって計算される。

「[集計](#)」、「[ディメンション](#)」、「[階層](#)」も参照。

エッジ(edge)

キューブまたはドキュメントにおいて、ともに表示される 1 つ以上の一連のディメンション。キューブのエッジ数に制限はないが、表示を目的としてデータが 3 つのエッジ（行エッジ、列エッジ、ページ・エッジ）に沿って構成される場合が多い。

クロス集計レポートでは、行エッジのディメンション・メンバーが最初の列に表示され、行を識別する。列エッジのディメンション・メンバーは最初の行に表示され、列を識別する。ページ・エッジのディメンション・メンバーは、レポートの個々のページのラベルとなる。

「[キューブ](#)」も参照。

オブジェクト(object)

アナリティック・ワークスペースのワークスペース・ディクショナリ内の個々の項目。アナリティック・ワークスペースは、変数、式、ディメンション、リレーション、プログラムなどの 1 つ以上のオブジェクトで構成されており、データを編成、格納および取得するために

これらのオブジェクトが使用される。各オブジェクトは、特定のオブジェクト型として作成され、特定のタイプの情報を格納する。同じ型のオブジェクト（たとえば、3つの変数）であっても、アナリティック・ワークスペース内で異なるロールを持つことができる。

「[ロール](#)」も参照。

オブジェクト型(object type)

Oracle のオブジェクト・テクノロジーにおけるユーザー定義のデータ型の1つで、実在するエンティティを抽象化したもの。オブジェクト型は、次の構成要素を持つスキーマ・オブジェクトである。

- 名前。スキーマ内でオブジェクト型を一意に識別する。
- 属性。実在するエンティティの構造および状態をモデル化する。
- メソッド。実在するエンティティの振る舞いを、PL/SQL または Java で実装する。

親(parent)

階層内で、特定のメンバーの直上のレベルにあるディメンション・メンバー。ディメンション階層においては、親のデータ値は、その子のデータ値の集計された合計である。

「[子](#)」と対比。「[階層](#)」、「[レベル](#)」も参照。

親子リレーション(parent-child relation)

階層ディメンションにおける、1つの親と1つ以上の子の間の1対多関係。たとえば、New York（州レベル）は、Albany、Buffalo、Poughkeepsie および Rochester（都市レベル）の親である。

「[子](#)」、「[親](#)」も参照。

親リレーション(parent relation)

各ディメンション・メンバーの親を格納することによってディメンションの階層を定義する、アナリティック・ワークスペースのリレーション・オブジェクト。

「[親](#)」、「[リレーション](#)」も参照。

オンライン・トランザクション処理(online transaction processing : OLTP)

高速かつ信頼性の高いトランザクション処理を行うために最適化されたシステム。ほとんどの OLTP 操作では、データ分析システムと比較して、扱う行の数は少なく、表のグループは大規模になる。

オンライン分析処理(online analytical processing : OLAP)

履歴データを動的に多次元分析する機能。次のような処理がサポートされる。

- ディメンションおよび階層にまたがる計算
- 傾向の分析

- 階層内のドリルアップおよびドリルダウン
- ディメンションの向きを変更するためのキューブの回転

解決済データ (solved data)

導出データがすべて計算済である結果セット。結果セットのすべてのデータは SQL ベースのアプリケーションに戻される前に計算されているので、アナリティック・ワークスペースからフェッチしたデータは常に完全に解決済である。アナリティック・ワークスペースから取得した結果セットは、そのデータが事前計算されたものであるか、その場で計算されたものであるかに関係なく同一である。

「[即時計算](#)」、「[事前計算](#)」も参照。

階層 (hierarchy)

データを編成する手段として順序付けされたレベルを使用する論理構造。階層は、データ集計を定義するために使用できる。たとえば、時間ディメンションでは、階層を使用して月レベルから四半期レベル、年レベルへとデータを集計できる。また、階層内のレベルが集計された合計を表しているかどうかに関係なく、ナビゲーション・ドリル・パスを定義するために階層を使用することもできる。

加算的 (additive)

加算で集計可能なファクト（またはメジャー）を指す。ファクトのタイプとしては加算的なファクトが最も一般的。例としては、売上、費用、利益などがある。

「[非加算的](#)」、「[準加算的](#)」と対比。

カスタム・メジャー (custom measure)

実行時に計算され、結果セットに追加される 1 つ以上のデータ列として表される導出メジャー。結果セットには、現在ステータスが設定されている各ディメンション・メンバーの値が含まれる。通常、カスタム・メジャーでは、1 つ以上のストアド・メジャーに対して計算を実行する単一行ファンクションを使用する。たとえば、分析者は、COSTS メジャーに対して OLAP DML の LAGDIF ファンクションを実行して前の期間からの費用の変化を計算するカスタム・メジャーを作成できる。または、SALES メジャーから COSTS メジャーを減算して利益を計算するカスタム・メジャーを作成できる。

「[ディメンション・メンバー](#)」、「[OLAP DML](#)」、「[メジャー](#)」、「[ステータス](#)」も参照。

カスタム・メンバー (custom member)

実行時に作成され、1 つ以上の既存のディメンション・メンバーの親として定義されるディメンションのメンバー。カスタム・メンバーのメジャーの値は、そのディメンションの集計規則を使用して計算される。

「[集計](#)」、「[ディメンション・メンバー](#)」、「[親](#)」も参照。

キューブ(cube)

同じディメンションを持つメジャーの論理的な編成。キューブのエッジにはディメンション・メンバーが格納され、キューブの本体にはデータ値が格納される。たとえば、売上データをキューブに編成する場合、エッジには時間、製品および顧客の各ディメンションの値が格納され、本体には売上数量および売上高のデータが格納される。スター・スキーマでは、キューブはファクト表で表される。

子(child)

階層内で、特定の値の直下のレベルにある値のこと。たとえば、Time ディメンションでは、値 Jan-02 は値 Q1-2002 の子である。子値が複数の階層に属している場合は、1 つの値が複数の親の子になることもある。

「[親](#)」と対比。「[子孫](#)」、「[階層](#)」、「[レベル](#)」も参照。

更新ウィンドウ(update window)

データベース内のデータを更新するために費やすことのできる時間の長さ。

コンテナ(container)

「[オブジェクト](#)」を参照。

コンポジット(composite)

データが存在するディメンション値の組合せ（タプルとも呼ばれる）をリストするアナリティック・ワークスペース・オブジェクト。コンポジットによってディメンション化された変数にデータ値が追加されると、これがトリガーとなってコンポジット・タプルが作成される。コンポジットは、1 つ以上のスパースなデータ変数への索引であり、スパースなデータを稠密に格納するために使用される。

「[ディメンション](#)」、「[スパース性](#)」、「[変数](#)」も参照。

サマリー(summary)

「[集計](#)」、「[マテリアライズド・ビュー](#)」を参照。

式(formula)

ワークスペース・オブジェクトの一種で、値を生成する格納済の計算、格納済の式またはストアド・プロシージャを表す。式は、複雑または使用頻度の高いデータ内の関係を、結果セットを格納せずに定義および保存する手段を提供する。式に対して問合せを発行するたびに、値を生成するために必要な計算またはプロシージャを OLAP エンジンが実行する。

事前計算(precaculate)

データ・メンテナンス手順としてデータを計算および格納すること。アナリティック・ワークスペースでは、集計データは、事前に計算するか、その場で計算するか、またはその両方を組み合わせて計算できる。

「[即時計算](#)」と対比。

子孫(descendant)

階層内で、特定のディメンション・メンバーよりも下位のレベルにあるディメンション・メンバー。直下のレベルのことは「子」という。

「祖先」と対比。「集計」、「子」、「階層」、「レベル」も参照。

集計(aggregation)

複数のデータ値を1つの値に集約するプロセス。たとえば、売上データを毎日収集して、これを週レベルに集計したり、週データを月レベルに集計したりできる。この結果のデータは、集計データとして参照できる。「集計」は「サマリー」と同義であり、「集計データ」は「サマリー・データ」と同義である。

修飾データ参照(qualified data reference)

1つ以上のディメンションを、OLAP DML コマンドの間だけ1つの値に制限する修飾子。QDR は、現在のステータスに影響を与えることなく1つの値を一時的に参照したい場合に有用。次の OLAP DML コマンドの例では、QDR は MONTH ディメンションを JUN02 に制限する。

```
SHOW sales(month 'JUN02')
```

「ディメンション」、「ディメンション・メンバー」、「ステータス」も参照。

準加算的(semi-additive)

すべてのディメンションではなく一部のディメンションに対する加算で集計可能なファクト（またはメジャー）を指す。例としては、社員数や在庫数などがある。

「加算的」、「非加算的」と対比。

スキーマ(schema)

関連するデータベース・オブジェクトの集まり。リレーショナル・スキーマはデータベース・ユーザー ID によってグループ化され、表やビューなどのオブジェクトを含む。多次元スキーマはアナリティック・ワークスペースと呼ばれ、ディメンション、リレーション、変数などのオブジェクトを含む。

「アナリティック・ワークスペース」、「スノーフレーク・スキーマ」、「スター・スキーマ」も参照。

スター・クエリー(star query)

1つのファクト表と複数のディメンション表との結合。各ディメンション表は、主キーから外部キーへの結合を使用してファクト表に結合される。ただし、ディメンション表同士は結合されない。

スター・スキーマ(star schema)

多次元データ・モデルを表すように設計されたリレーショナル・スキーマ。スター・スキーマは、1 つ以上のファクト表と、外部キーを通じて関連付けられた 1 つ以上のディメンション表で構成される。

「スキーマ」、「スノーフレーク・スキーマ」も参照。

スタンダード・フォーム(standard form)

「データベース・スタンダード・フォーム」を参照。

ステータス(status)

特定のディメンションに関する、現在アクセス可能な値のリスト。特定のディメンションのステータスが、そのディメンションの格納済の値のサブセットに制限されている場合、そのディメンションに基づくすべての式が、対応するデータのサブセットに制限される。ディメンションのステータスは特定のセッションを通じて維持され、明示的に変更しないかぎりには変更されない。アナリティック・ワークスペースが最初にセッションにアタッチされたときは、すべてのメンバーにステータスが設定される。

「ディメンション」、「ディメンション・メンバー」も参照。

スノーフレーク・スキーマ(snowflake schema)

ディメンション表が部分的または完全に正規化された、スター・スキーマの一種。

「正規化」、「スキーマ」、「スター・スキーマ」も参照。

スパース性(sparsity)

実際のデータを持たないディメンション値の組合せが比較的高い割合で存在するような多次元データを指す概念。このような「空の」値（NA 値）であっても、アナリティック・ワークスペースの記憶域を占有する。コンボジットを作成すると、スパースなデータを効率的に扱うことができる。

スパース性には、次の 2 つの種類がある。

- 1 つ以上のディメンションの一定範囲の値がデータを持たない場合に発生する制御されたスパース性。たとえば、新しい変数を月でディメンション化したが、過去の月のデータは存在しないという場合がこれに該当する。月ディメンションには過去の月があることでセルは存在するが、そのセルには NA 値が格納される。
- NA 値が変数全体に点在している場合に発生するランダムなスパース性。通常、これはディメンション値の一部の組合せが決してデータを持たないことが原因で発生する。たとえば、ある地区では特定の製品のみを販売しており、販売していない製品についてのデータは持っていないが、別の地区ではそうした製品を販売しているという場合がこれに該当する。

「コンボジット」、「NA 値」も参照。

正規化(normalize)

リレーショナル・データベースにおいて、データを複数の表に分離することによりデータの冗長性を取り除くプロセス。「[非正規化](#)」と対比。

セル(cell)

式の単一のデータ値。ディメンション化された式の場合、セルは、式の各ディメンションの1つの値によって識別される。たとえば、MONTH ディメンションと DISTRICT ディメンションを持つ変数の場合、月と地区の各組合せによって、その変数の個々のセルが識別される。

「[ディメンション](#)」、「[変数](#)」も参照。

ソース(source)

データウェアハウスのデータの抽出元となるデータベース、アプリケーション、ファイルまたはその他の記憶域機構。

即時計算(on-the-fly)

特定の問合せに対する応答において、実行時に計算を行うこと。アナリティック・ワークスペースでは通常、カスタム・メジャーおよびカスタム・メンバーはその場で計算される。集計データは、事前に計算するか、その場で計算するか、またはその両方を組み合わせて計算できる。

「[事前計算](#)」と対比。

属性(attribute)

単一のディメンション・メンバーまたはディメンション・メンバーのグループの説明的な特性。単一のメンバーに割り当てられた場合、属性は、表示用（説明的な名前など）または分析（期間における日数など）に使用できる補足的な情報を提供する。グループに割り当てられた場合は、同じような特性に基づいてデータを選択できるようにする論理的な分類を表す。たとえば、履物のデータベースでは、同じ色のブーツ、スニーカーおよびスリッパをすべて選択するのに、色の属性を使用できる。

祖先(ancestor)

階層内で、特定の値よりも上位のレベルにある値。たとえば、Time ディメンションでは、値 2002 は値 Q1-02 と Jan-02 の祖先である。ディメンション階層においては、祖先のデータ値はその子孫のデータ値を集計した値である。

「[子孫](#)」と対比。「[階層](#)」、「[レベル](#)」、「[親](#)」も参照。

抽象データ型(abstract data type : ADT)

「[オブジェクト型](#)」を参照。

データ・ウェアハウス(data warehouse)

トランザクション処理用ではなく、問合せおよび分析用に設計されたリレーショナル・データベース。通常、データ・ウェアハウスには、トランザクション・データから導出された履

歴データが格納されるが、別のソースのデータが格納される場合もある。データ・ウェアハウスにより、分析ワークロードとトランザクション・ワークロードを分離できる。また企業は、複数のソースのデータを統合できるようになる。

データソース(data source)

データを提供するデータベース、アプリケーション、リポジトリまたはファイル。

データベース・スタンダード・フォーム(database standard form)

階層ディメンション、レベル・ディメンション、親リレーション、レベル・リレーションなど、特定のオブジェクトの集まりで構成されたアナリティック・ワークスペース。各オブジェクトには、そのロールおよびアナリティック・ワークスペース内の他のオブジェクトとの関係を識別する一連のプロパティが定義されている必要がある。スタンダード・フォームは、OLAP ツールでアナリティック・ワークスペースにアクセスできるようにするために必要だが、多次元分析に必須というわけではない。

定義(definition)

アナリティック・ワークスペース・オブジェクトを記述するもの。オブジェクト定義には、オブジェクトの名前、タイプ（ディメンションや変数など）、データ型、ディメンション、詳細な説明、権限仕様、プロパティなどの特性が含まれる。

「[ディクショナリ](#)」、「[オブジェクト](#)」、「[プロパティ](#)」も参照。

ディクショナリ(dictionary)

アナリティック・ワークスペース内のオブジェクト定義の集まり。ディクショナリは、ワークスペース・ディクショナリとも呼ばれる。

「[定義](#)」、「[オブジェクト](#)」も参照。

ディメンション(dimension)

データを分類する構造。売上に関するデータで最も一般的なディメンションは、時間、地理および製品である。ほとんどのディメンションは階層を持つ。

アナリティック・ワークスペースにおけるディメンションは、値のリストのコンテナである。ディメンションは、変数の値を識別する索引として機能する。たとえば、売上データが月ごとの売上高を個別に持っている場合、そのデータは月ディメンションを持っていることになる。つまり、データは月別に編成されていることになる。

SQL におけるディメンションは、列セットの組の階層関係（親子関係）を定義する一種のオブジェクトである。

「[階層](#)」も参照。

ディメンション値(dimension value)

「[ディメンション・メンバー](#)」を参照。

ディメンション・ビュー(dimension view)

スター・スキーマのディメンション表と同じタイプのデータ、つまりディメンション・メンバーおよび属性の列を格納する、アナリティック・ワークスペース内のデータのリレーショナル・ビュー。通常、ディメンション・ビューは、ディメンション階層のレベルに関係なく、キー列のすべてのディメンション・メンバーを示す。

「[ディメンション表](#)」、「[スター・スキーマ](#)」も参照。

ディメンション表(dimension table)

スター・スキーマまたはスノーフレーク・スキーマの論理ディメンションのすべての値またはその一部を格納するリレーショナル表。ディメンション表は、時間、部門、所在地、製品などの階層情報および分類情報として表される企業のビジネス・エンティティを記述する。ディメンション表は、ルックアップ表または参照表とも呼ばれる。

ディメンション・メンバー(dimension member)

ディメンションを構成するリストの一要素。ディメンション値とも呼ばれる。たとえば、コンピュータ・メーカーの製品ディメンションには、LAPPC や DESKPC というディメンション・メンバーが含まれる。また地理ディメンションには、Boston や Paris などのメンバーが含まれる。時間ディメンションには、NOV02、DEC02、JAN03、FEB03、MAR03 などのメンバーが含まれる。

導出ファクト / メジャー(derived fact (or measure))

算術演算またはデータ変換を使用して既存のデータから生成されたファクト（またはメジャー）。例としては、平均、合計、割合、差などがある。

ドリル(drill)

1 つの項目から一連の関連項目にナビゲートすること。通常、ドリル操作では、階層内のレベルを上方向または下方向にナビゲートする。データを選択する際、階層内でドリルダウンまたはドリルアップすることで、階層を開いたり閉じたりできる。

ドリルアップ(drill up)

階層内の親値に関連付けられている子孫の値のリストを閉じること。

ドリルダウン(drill down)

ビューを開いて、階層内の親値に関連付けられている子値を表示すること。

非加算的(nonadditive)

平均など、加算で集計できないファクト（またはメジャー）を指す。「[加算的](#)」、「[準加算的](#)」と対比。

非正規化(denormalized)

表の中に冗長性を許容すること。「[正規化](#)」と対比。

ファクト(fact)

「[メジャー](#)」を参照。「[加算的](#)」、「[導出ファクト/メジャー](#)」も参照。

ファクト表(fact table)

ファクトを格納する、スター・スキーマ内の表。多くの場合、ファクト表には、ファクトを格納する列と、ディメンション表への外部キーとなる列の2種類の列がある。通常、ファクト表の主キーは、その表のすべての外部キーで構成されるコンボジット・キーである。

ファクト表には、ディテール・レベルのファクトと集計済のファクトのどちらでも格納できる。集計済のファクトを格納するファクト表は通常、集計表またはマテリアライズド・ビューと呼ばれる。通常、ファクト表には同じ集計レベルのファクトが格納される。

ファミリ・リレーション(family relation)

各ディメンション・メンバーの完全な親子関係を識別する、アナリティック・ワークスペースのリレーション・オブジェクト。ファミリ・リレーションは、データ・ディメンションとレベル・ディメンションという少なくとも2つのディメンションを持つ。リレーションの内容によって、ディメンション・メンバーごとに階層の各レベルでの祖先が識別される。

「[祖先](#)」、「[レベル](#)」、「[リレーション](#)」も参照。

プログラム(program)

データベース・オブジェクトの一種で、一連の OLAP DML コマンドを含む。プログラムは、関連する一連のコマンドを実行する。プログラムをネストして、1つのプログラムで別のプログラムをコールすることもできる。プログラムは値を戻すことが可能で、この場合はユーザー定義ファンクションと呼ばれる。

「[オブジェクト](#)」も参照。

プロパティ(property)

オブジェクトまたは構成要素の特性。プロパティは、識別子および説明を提供したり、オブジェクトの特徴（小数点以下の桁数や色など）を定義したり、オブジェクトの振る舞い（オブジェクトが有効かどうかなど）を定義したりする。プロパティは、スタンダード・フォームのアナリティック・ワークスペースで幅広く使用される。

「[オブジェクト](#)」も参照。

ベース・レベル・データ(base level data)

最低レベルのデータ。トランザクション・データベースなどの別のソースから取得されることが多い。

「[集計](#)」と対比。

変数(variable)

データを格納する、ワークスペース・オブジェクトの一種。変数のデータ型は、その変数にどのような種類のデータ（数値やテキスト・データなど）が格納されるかを示す。

変数がディメンションを持つ場合、それらのディメンションがその変数のデータを編成する。ディメンション・メンバーの組合せごとに、セルが1つある。ディメンション化された変数は配列であり、そのセルが個々のデータ値である。ディメンションを持たない変数は単一セルの変数であり、1つのデータ値を格納する。

「セル」、「ディメンション」、「ディメンション・メンバー」、「オブジェクト」も参照。

マッピング(mapping)

ソース・オブジェクトとターゲット・オブジェクト間の関係およびデータ・フローの定義。

マテリアライズド・ビュー(materialized view)

ファクト表およびディメンション表（ファクト表のみの場合もある）のデータを集計または結合したデータで構成される事前計算済のリレーショナル表。サマリー表または集計表とも呼ばれる。

メジャー(measure)

売上データや費用データなど、調査および分析が可能なデータ。メジャー内のデータは、選択して表示することができる。メジャーは、変数またはリレーションとして格納したり、式を使用して計算することができる。メジャーとファクトは同義語である。一般的に、メジャーは多次元環境で使用され、ファクトはリレーショナル環境で使用される。

メジャーには、ベース・メジャーとカスタム・メジャーがある。ベース・メジャー（売上数量や売上高など）は格納される。カスタム・メジャー（昨年のシェア率など）はベース・メジャーから計算される。

「式」、「リレーション」、「変数」も参照。

メジャー・ビュー(measure view)

スター・スキーマのファクト表と同じタイプのデータを格納する、アナリティック・ワークスペース内のデータのリレーショナル・ビュー。ただし、メジャー・ビューには、ベース・レベルのファクトの他に、集計や行間計算などの導出データも格納される。

「ファクト表」、「スター・スキーマ」も参照。

メタデータ(metadata)

データおよびその他の構造（オブジェクト、ビジネス・ルール、ビジネス・プロセスなど）を記述するデータ。

「OLAP カタログ」も参照。

モデル(model)

アナリティック・ワークスペース・オブジェクトの一種。データを計算し、それを変数またはディメンション値に割り当てるために使用される、相互に関連のある一連の方程式を格納する。モデルは、財務データを扱う場合によく使用される。

「ディメンション・メンバー」、「オブジェクト」、「変数」も参照。

リレーション(relation)

ワークスペース・オブジェクトの一種。変数と似ているが、データ値を特定のデータ型 (NUMBER など) ではなく特定のディメンション (PRODUCT など) のメンバーに限定する点異なる。リレーションは、特定のディメンションの値と、同じディメンションまたはデータベース内の別のディメンションの値との対応関係を確立する。

たとえば、都市と販売地域の間には、各都市が特定の地域に属しているというリレーションが存在する。都市と販売地域の間のリレーションにおいては、リレーションは CITY でディメンション化される。各セルは、REGION ディメンションの対応する値を保持する。

「セル」、「ディメンション」、「ディメンション・メンバー」、「変数」も参照。

レベル(level)

階層内での位置。たとえば、時間ディメンションは、月レベル、四半期レベルおよび年レベルのデータを表す階層を持つ。

レベル・リレーション(level relation)

各ディメンション・メンバーのレベルを識別する、アナリティック・ワークスペースのリレーション・オブジェクト。

「レベル」、「リレーション」も参照。

ロール(role)

ワークスペース・オブジェクトが属するオブジェクト型のカテゴリにおける、そのオブジェクトの役割。たとえば、数値のビジネス・メジャーを格納する変数は、メジャーというロールを持つ。製品の説明的な名前を格納する変数は、属性というロールを持つ。どちらも変数だが、それぞれ異なるタイプの情報を格納しており、多次元モデルにおいて異なる役割を果たす。

「オブジェクト」も参照。

ロールアップ形式(rollup form)

各ディメンション・メンバーの完全な系図を 1 つの行に表示する表。この表は、階層の各レベルに対応する列を持つ。

たとえば、ベース・レベルのディメンション・メンバーである Florence の行は、City 列に FLORENCE、Country 列に ITALY、Region 列に EUROPE を持つ。Italy の行は、City 列に NULL、Country 列に ITALY、Region 列に EUROPE を持つ。

「埋込み合計」と対比。「祖先」、「ディメンション・メンバー」、「階層」も参照。

A

aggmap
「集計マップ」を参照
AGGREGATE ファンクション
式, 9-9
ALL_ATTRIBUTES ディメンション, 8-18, C-27
ALL_ATTRTYPES ディメンション, C-29
ALL_CUBES ディメンション, 8-25, C-24
ALL_DESCRIPTIONS 変数, 8-6, 8-12, 8-16, 8-18,
8-21, 8-25, 9-13, C-34
ALL_DESCTYPES ディメンション, C-29
ALL_DIMENSIONS ディメンション, 8-6, C-25
ALL_HIERARCHIES ディメンション, 8-11, C-25
ALL_LANGUAGES ディメンション, 8-17, C-30
ALL_LEVELS ディメンション, 8-16, C-26
ALL_MEASURES ディメンション, 8-20, 9-12, C-25
ALL_OBJECTS ディメンション, C-27
ALL_OBJTYPES ディメンション, C-28
ALL_OLAP2_AW ビュー, 7-3
ALTER SESSION コマンド, 12-9
Analytic Workspace Manager, 6-1 ~ 6-32
ATTR_INHIER 変数
データベース・スタンダード・フォーム, C-35
ATTRDEF オブジェクト, C-23
AW\$CLASS プロパティ, C-8
AW\$CREATEDBY プロパティ, C-8
AW\$LASTMODIFIED プロパティ, C-8
AW\$LOADPRGS プロパティ, 6-30
AW\$ROLE プロパティ, C-8 ~ C-23
AW\$STATE プロパティ, C-8
AW\$ 表, 12-12
AW_COMPSPECS 変数, C-33
AW_LOOPSPECS 変数, C-34
AW_NAMES 変数, 8-6, 8-18, 8-21, 8-25, 9-15,

C-33

AW_ROLES ディメンション, C-29

B

BFILE セキュリティ, 12-10
BI Beans
アナリティック・ワークスペースの有効化, 6-24
定義済, 4-2, 4-3

C

CHANNEL_DIM 表
定義済, 3-13
CHARSET オプション, A-3
COMMIT コマンド, A-6
COMSPEC オブジェクト, C-17
CONNECT ロール, 12-6
CREATE_DB_STDFORM プログラム, A-6
Crosstab Bean, 4-5
Cube Viewer, 5-11
CUBE_MEASURES 値セット, 8-21, 8-26, 9-15, C-31
CUBEDEF ディメンション, C-3, C-15
CUST_MEAS 列, 7-7
CUSTOMER_DIM 表
定義済, 3-11
CWM2
書込み API, 5-14
定義済, 1-11
CWM2_OLAP_CATALOG パッケージ, 5-15
CWM2_OLAP_CUBE パッケージ, 5-14
CWM2_OLAP_DIMENSION パッケージ, 5-14
CWM2_OLAP_HIERARCHY パッケージ, 5-14
CWM2_OLAP_LEVEL_ATTRIBUTE パッケージ, 5-14
CWM2_OLAP_LEVEL パッケージ, 5-14

CWM2_OLAP_MEASURE パッケージ, 5-15
CWM2_OLAP_PC_TRANSFORM パッケージ, 5-15
CWM2_OLAP_VALIDATE パッケージ, 5-15

D

DB_CACHE_SIZE パラメータ, 12-7
DBMS_AW_UTILITIES パッケージ
 カスタム・メジャーの管理, 7-4
 定義済, 1-11, 7-2
DBMS_AWM パッケージ
 定義済, 1-11, 7-2
DBMS_AW パッケージ
 EXECUTE プロシージャ, 9-6
 OLAP DML コマンドの実行, 9-3
 カスタム・メジャーの管理, 7-4
 定義済, 1-11, 7-2
DEFAULT_HIER リレーション, 8-12, C-35
DIM_ATTRIBUTES 値セット, C-32
DIM_HIERARCHIES 値セット, 8-12, C-31
DIM_LEVELS 値セット, 8-6, 8-16, C-32
DIMDEF ディメンション, C-3, C-18
Discoverer
 アナリティック・ワークスペースの有効化, 6-26

E

EDDE.HIERMNT プログラム (廃止), A-12
EDDE.MSG プログラム (廃止), A-11
EEX ファイル
 アナリティック・ワークスペース用に生成, 6-26
EIF ファイル, A-6
End User Layer (EUL)
 アナリティック・ワークスペース用に作成, 6-26
Express Connection Utility (廃止), A-4
Express Relational Access Administrator (廃止), A-3
Express Relational Access Manager (廃止), A-3
EXTCALL (廃止), A-5

F

familyrel リレーション
 データベース・スタンダード・フォーム, 8-14

G

Global Computing Company

 データ要件, 3-1 ~ 3-7
GLOBAL_AW ユーザー
 定義, 6-11
GLOBALX 表, B-2
Global スター・スキーマ
 定義済, 3-8 ~ 3-14
Graph Bean, 4-4

H

HIER_LEVELS 値セット, C-22
HIERLEVELS 値セット, C-3
HIERLIST ディメンション, C-3, C-19
 属性, 8-16
 データベース・スタンダード・フォーム, 8-7

I

IDE
 定義済, 4-2
inhier 変数
 データベース・スタンダード・フォーム, 8-10
init.ora ファイル, 12-7

J

Java
 サンドボックス・セキュリティ, 4-2
 定義済, 4-1

L

LAG ファンクション
 OLAP DML, 7-18
LEVELLIST ディメンション, C-3, C-20
 データベース・スタンダード・フォーム, 8-12
LEVELREL リレーション, 8-13, C-3, C-21
LIMIT コマンド, 9-13
LOOPSPEC コンボジット, C-16

M

MDI
 定義済, 4-3
MEASUREDEF オブジェクト, C-3, C-17
MEMBER_CREATEDBY 変数, C-37
MEMBER_FAMILYREL リレーション, C-37

MEMBER_INHIER 変数, C-36
MR_REFRESH プロシージャ, 7-26

N

NLS_LANG 構成パラメータ, A-3

O

ODBC サポート (廃止), A-3
oescmd プログラム (廃止), A-2
oesmgr プログラム (廃止), A-2
OLAP
 定義済, 1-2
OLAP API
 アナリティック・ワークスペースの有効化, 6-24
 定義済, 1-4, 4-2, 4-6
OLAP Beans, 4-3, 4-5
OLAP DML
 OLAP Worksheet で実行, 9-4
 OLAP Worksheet での使用, 6-5
 SQL インタフェース, 7-2
 コマンドの実行, 9-3
 定義済, 1-4
 特殊記号, 9-21
 プログラムの編集, 9-5
OLAP Instance Manager (廃止), A-2
OLAP Server データベース
 スタンダード・フォームへの変換, A-6
OLAP Worksheet, A-4
 Analytic Workspace Manager で開く, 6-5
 セッション共有, 6-4
 定義済, 6-6
OLAP_PAGE_POOL_SIZE, 12-8
OLAP_SYS_LIMITMAP 変数, 7-17
OLAP_TABLE ファンクション, 7-11 ~ 7-25
OLAP カタログ
 CWM2 サンプル・スクリプト, 7-25 ~ 7-28
 アナリティック・ワークスペースに対する使用,
 5-1
 書込み API, 5-2
 サンプル・スクリプト, B-4
 定義済, 1-4
 メタデータ・モデル表, 5-2
 読込み API, 5-2
 リレーショナル表に対する使用, 5-2
OLAP 管理ツール, 5-8

OLAP メタデータ
 Analytic Workspace Manager における表示, 6-4
 CWM2 API での作成, 5-14
 Oracle Enterprise Manager での作成, 5-8
 作成ツール, 5-3
 作成手順, 5-3
 マテリアライズド・ビュー, 13-3
OLTP
 定義済, 1-2
Oracle JDeveloper, 4-2

P

PARALLEL_MAX_SERVERS パラメータ, 12-7
PARENTREL リレーション, C-3, C-21
parentrel リレーション
 データベース・スタンダード・フォーム, 8-8
Personal Express (廃止), A-4
pfile 設定, 12-7
PGA_AGGREGATE_TARGET パラメータ, 12-7
PGA の割当て, 12-8
PRICE_AND_COST_HISTORY_FACT 表
 定義済, 3-14
 ワークスペース・オブジェクトへのマッピング,
 3-18
PRICE_AND_COST_UPDATE_FACT 表
 定義済, 3-14
PRODUCT_DIM 表
 定義済, 3-12
 ワークスペース・オブジェクトへのマッピング,
 3-15
PS\$ 表, 12-12

Q

Query Builder, 4-6
QUERY REWRITE システム権限, 12-6

R

Relational Access Administrator (廃止), A-3
Relational Access Manager (廃止), A-3
ROLLUP コマンド, A-6, A-7

S

SEGWIDTH

「セグメント・サイズ」を参照
SELECT 権限, 12-7
SESSIONS パラメータ, 12-7
SNAPI 通信 (廃止), A-4
SPLExecutor クラス
 OLAP DML コマンドの実行, 9-3
SQL*Plus セッションの設定, 7-25
SQL インタフェース, 1-4
SQL コマンド (OLAP DML), A-2
SQL 分析関数, 4-7

T

Table Bean, 4-5
TIME_DIM 表
 定義済, 3-10
 ワークスペース・オブジェクトへのマッピング,
 3-16
Time 属性
 変換済 OLAP Server データベースに対して作成,
 A-11

U

Unicode, A-3
UNITS_HISTORY_FACT 表
 定義済, 3-13
UNITS_UPDATE_FACT 表
 定義済, 3-13
UPDATE コマンド, A-6

V

VISIBLE 変数, C-36

X

XCA (非サポート), A-4
XPDDDATA データベース (廃止), A-12

あ

アクセス権, 12-6
アクティブ・カタログ, 7-3
アナリティック・ワークスペース
 Discoverer に対する有効化, 6-24, 6-26
 OLAP API に対する有効化, 6-24

一般的な用途, 1-4
オブジェクト・リレーションシップ, 2-8
基本プロセスの概要, 1-5
作成の基本手順, 6-10
集計プラン, 6-20
スタンダード・フォーム, 6-7
定義済, 1-4
データベース記憶域, 12-11
メンテナンス, 6-30
アプリケーション
 OLAP Server との相違, A-4
アロケーション, A-5
アロケーション・コマンド
 OLAP DML, 7-6

う

ウィザード
 Analytic Workspace Manager, 6-11 ~ 6-30
 BI Beans, 4-5

え

演算子
 OLAP DML, 7-4

お

オブジェクト型, 7-11
 作成用の構文, 7-15
オブジェクト指向プログラミング, 4-6
オペレーティング・システムのコマンド (OLAP DML
 で廃止), A-5
親子リレーション
 定義済, 2-3
親リレーション
 「parentrel リレーション」を参照
 定義済, 2-11

か

カーソル, 4-7
回帰
 OLAP DML, 7-4, 10-1
階層
 アナリティック・ワークスペース, 2-10
 作成、論理, 5-10

- データベース・スタンダード・フォーム, 8-7
- 論理, 2-3
- 階層ディメンション
 - 「HIERLIST ディメンション」を参照
- 階層内変数
 - 「inhier 変数」を参照
- 拡張クラス
 - データベース・スタンダード・フォーム, C-39
- カスタム・メジャー
 - BI Beans のサポート, 4-6
 - DBMS_AW_UTILITIES の構文, 7-7
 - 管理, 7-3 ~ 7-11
 - マッピングのビュー, 7-7
- カタログ
 - AWCREATE, 8-30
 - OLAP API イネーブラ, 8-27
 - データベース・スタンダード・フォーム, 8-26
- カタログ・クラス
 - データベース・スタンダード・フォーム, C-24

き

- 起動パラメータ
 - データベース, 6-15
- 機能クラス
 - データベース・スタンダード・フォーム, C-34
- キャッシュ
 - 反復問合せでの使用, 4-7
- キャラクタ・セット, A-3
- キューブ
 - 定義済, 5-10
 - データベース・スタンダード・フォーム, 8-21
 - 表示, 5-11
 - 論理, 2-7
- キューブ・ディメンション
 - 定義済, 8-21
- キューブのリフレッシュ
 - 基本手順, 6-30

く

- クラス
 - データベース・スタンダード・フォーム, C-4
- グローバルゼーション, A-3

け

- 計算エンジン
 - 定義済, 1-4
- 結果セット, 4-7
- 結合, A-6
- 言語サポート, A-3

こ

- 構成手順, 12-1
- コンボジット
 - 定義, 6-8
- データベース・スタンダード・フォーム, 8-24

さ

- サーバー・パラメータ・ファイル, 12-7
- 最後に変化するディメンション, 6-8
- 最初に変化するディメンション, 6-8
- 最適化方法, 12-2
- 財務アプリケーション, 1-3
- 財務演算
 - OLAP DML, 7-5
- 作成オプション
 - アナリティック・ワークスペース, 6-10
- 参照整合性, 2-9

し

- 時間ディメンション, 5-10
- 式
 - アナリティック・ワークスペース, 2-12
 - 定義, 9-10
- 時系列ファンクション
 - OLAP DML, 7-4
- システム・プロパティ
 - データベース・スタンダード・フォーム, C-7
- 実行時集計, 6-19
- 実装クラス
 - データベース・スタンダード・フォーム, C-14
- 集計演算子
 - アナリティック・ワークスペース, 6-20
- 集計コマンド
 - OLAP DML, 7-6
- 集計データ
 - 事前計算, 6-19

- 即時, 6-19
- 集計プラン
 - 作成, 6-20
 - 定義済, 6-19
- 集計マップ
 - データベース・スタンダード・フォーム, 8-23
- 集計用バッチ・ウィンドウ, 6-19
- 修飾データ参照 (QDR), 9-14
- 需要計画システム, 1-3
- 初期化パラメータ, 12-9
- 書式設定
 - データ, 4-4
- 事例
 - 「Global Computing Company」を参照

す

- 数値書式設定, 4-4
- 数値ファンクション
 - OLAP DML, 7-5
- スキーマ
 - スター、スノーフレーク, 5-8
- スター・スキーマ
 - 定義済, 2-4
- スタンダード・フォーム
 - 「データベース・スタンダード・フォーム」を参照
- ストライプ化, 12-2
- スパース特性, 6-12, 6-17
- スポットライト書式設定, 4-4

せ

- 制限の指定
 - アナリティック・ワークスペース, 6-8
- 制限マップ
 - OLAP_TABLE ファンクション, 7-17
 - アナリティック・ワークスペースに格納, 7-17
- セグメント・サイズ
 - 基本的な規則, 6-9
- セッション共有, A-4
- 接続文字列
 - Analytic Workspace Manager, 6-7

そ

- 属性
 - アナリティック・ワークスペース, 2-12

- 作成、論理, 5-10, 5-14
- データベース・スタンダード・フォーム, 8-16
- 「レベル属性」または「ディメンション属性」も参照
- 論理, 2-4

た

- 代入文, 9-15
- タプル
 - コンボジット, 6-8

ち

- 抽象データ型
 - 「オブジェクト型」を参照

て

- ディメンション
 - アナリティック・ワークスペース, 2-9
 - 関係, 2-5
 - 作成、論理, 5-10
 - 時間, 5-10
 - データベース・スタンダード・フォーム, 8-3
 - 論理, 2-3, 5-9
- ディメンション階層
 - 「階層」を参照
- ディメンションの順序
 - 基本的な規則, 6-9
- ディメンション表
 - 定義済, 2-5
 - メタデータの定義, 5-9
- ディメンション・メンバー
 - 選択, 9-13
- ディレクトリ
 - データベース, 12-9
- データ書式設定, 4-4
- データ・ディクショナリ
 - 「アクティブ・カタログ」を参照
- データのストライプ化, 12-2
- データベース構成, 12-1
- データベース・スタンダード・フォーム
 - 拡張, 8-3
 - 基本, 9-1
 - 仕様, C-1 ~ C-39
 - 新規メジャーの作成, 9-8

- 定義済, 8-2
- データベース・セキュリティ, 12-5
- データ・モデル, 2-1
- データ・リフレッシュ, 6-30
- テーブル・ファンクション
 - 定義済, 7-12
- テキスト操作
 - OLAP DML, 7-5

と

- 統計演算
 - OLAP DML, 7-5
- 動的パフォーマンス表, 12-13
- ドリル, 4-4

に

- 認証, 12-5

ね

- ネーミング規則
 - データベース・スタンダード・フォーム, C-4

は

- パフォーマンス・カウンタ, 12-13
- パラメータ・ファイル, 12-7

ひ

- ビボット, 4-4
- ビュー
 - Discoverer に対して作成, 6-28
 - OLAP_TABLE ファンクションで作成, 7-17
- 表型
 - 作成用の構文, 7-15
- 表領域
 - アナリティック・ワークスペース, 12-2
 - 定義, 6-16, B-1

ふ

- ファイル
 - アクセス許可, 12-9
- ファクト表

- 定義済, 2-6
- メタデータの定義, 5-10
- ファミリー・リレーション
 - 「familyrel リレーション」を参照
- プレゼンテーション Beans, 4-3
- プログラム
 - OLAP DML の編集, 9-6
 - OLAP DML を実行, 9-5

へ

- ページ・プール
 - Oracle OLAP, 12-8
- ページング, 4-4
- 変換表, A-3
- 変数
 - アナリティック・ワークスペース, 2-11
 - セルの指定, 9-14

ま

- マテリアライズド・ビュー
 - CWM2, 13-6
 - グルーピング・セット, 13-6 ~ 13-14
 - 定義済, 2-6
- マルチバイト・キャラクタ・セット
 - OLAP Server と同等, A-3

め

- メジャー
 - アナリティック・ワークスペース, 2-11
 - カスタム, 4-6
 - 関係, 2-6
 - データベース・スタンダード・フォーム, 8-19
 - 論理, 2-2

も

- 文字列ファンクション
 - OLAP DML, 7-5
- モデリング・コマンド
 - OLAP DML, 7-6
- モデリング・サポート, A-5

ゆ

有効化

OLAP API のビューの作成, 6-24

OLAP カタログ・メタデータの作成, 6-25

アナリティック・ワークスペース, 6-24

ユーザー・アクセス

サンプル・スクリプト, B-1

ユーザーのアクセス権, 12-6

ユーザー名, 12-5

ロール, 12-6

ログイン名, 12-5

ログ・ファイル

Analytic Workspace Manager における生成, 6-11

よ

予測コマンド

OLAP DML, 7-4, 10-1

予測分析アプリケーション, 1-3

ら

ランク書式設定, 4-4

り

リフレッシュ・プロセス

アナリティック・ワークスペース・データ, 6-30

リレーション

アナリティック・ワークスペース, 2-10

れ

レベル

アナリティック・ワークスペース, 2-10

作成、論理, 5-10

データベース・スタンダード・フォーム, 8-12

論理, 2-3

レベル・ディメンション

「levellist ディメンション」を参照

レベル・リレーション

「levelrel リレーション」を参照

定義済, 2-11

連立方程式, 7-6

ろ

ローカライゼーション, A-3

ロード・プログラム, 6-30

編集, 6-31